

SIMPLICIAL DECOMPOSITION AND DUAL
METHODS FOR NONLINEAR NETWORKS

By

JOSE A. VENTURA

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL OF
THE UNIVERSITY OF FLORIDA IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1986

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude and appreciation to the chairman of my supervisory committee, Dr. D.W. Hearn, for his continuous interest, guidance and patience throughout this research work. It is a pleasure to thank the other members of my supervisory committee, Dr. H.P. Benson, Dr. D.J. Elzinga, Dr. R.L. Francis and Dr. H.W. Hamacher, for their assistance.

I would also like to express my appreciation to Dr. R. Andreu, Dr. J. Ribera and, especially, Dr. J. Riverola of I.E.S.E. (Barcelona, Spain), for their early confidence in me and for giving me the encouragement to pursue a Ph.D.

Words cannot tell how much I owe my wife Marta, my daughter Martita, my parents and my sister. To them I dedicate this dissertation.

This research was partially supported by the National Science Foundation under NSF Grants ECE-8420830 and ECS-8516365.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	ii
ABSTRACT	v
CHAPTER	
ONE INTRODUCTION AND OVERVIEW.	1
1.1 Introduction	1
1.2 Overview	3
TWO PROBLEM STATEMENT AND PAST WORK.	6
2.1 Problem Statement.	6
2.2 Past Work.	7
2.2.1 Frank-Wolfe Method	7
2.2.2 Convex Simplex Method.	8
2.2.3 Reduced Gradient Method.	9
2.2.4 Piecewise-linear Approximation Method.	11
2.2.5 Primal Truncated Newton Method	13
2.2.6 Relaxation Method.	14
THREE RESTRICTED SIMPLICIAL DECOMPOSITION.	17
3.1 Background	17
3.2 Preliminaries.	18
3.3 Restricted Simplicial Decomposition and Global Convergence.	21
3.4 Finite Convergence	25
3.5 Master Problem	32
3.5.1 Bertsekas Projection Method.	34
3.6 Implementation	39
3.7 Computational Results.	48
3.7.1 Colville Problems.	48
3.7.2 Traffic Assignment Problems.	49
3.7.3 Electrical Network Problems.	57
3.7.4 Water Distribution Problems.	58

	Page
FOUR DUAL METHODS FOR SEPARABLE STRICTLY CONVEX QUADRATIC NETWORK PROBLEMS	62
4.1 Introduction	62
4.2 CASE I: Networks with Undirected Arcs.	63
4.2.1 Conjugate Gradient Method.	64
4.2.2 Computational Experiments.	70
4.3 CASE II: Networks with Upper and Lower Bounds on the Arcs	71
4.3.1 Dual Ascent Method	78
4.3.1.1 Computation of d_k	79
4.3.1.2 Computation of α^k	80
4.3.2 Examples	86
FIVE SUMMARY.	90
APPENDIX	
A RSD FOR AN UNBOUNDED FEASIBLE REGION	92
B RSD FOR NONLINEAR CONSTRAINTS.	95
C FURTHER COMPUTATIONAL NOTES.	100
REFERENCES	102
BIOGRAPHICAL SKETCH.	109

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

SIMPLICIAL DECOMPOSITION AND DUAL
METHODS FOR NONLINEAR NETWORKS

By

JOSE A. VENTURA

August, 1986

Chairman: Dr. D.W. Hearn

Co-Chairman: Dr. D.J. Elzinga

Major Department: Industrial and Systems Engineering

This study is concerned with the development of algorithms for minimizing a nonlinear objective function subject to linear constraints. Special attention is given to models that have network constraints with possible bounds on the variables because they have many applications in areas such as transportation science, engineering and economics.

The first approach studied is a restricted version of the simplicial decomposition algorithm. This method allows the user to treat the maximum size of the generated simplices as a parameter. When the parameter is at its minimum value, the method reduces to the Frank-Wolfe algorithm; when at its maximum, it is the original simplicial decomposition of von Hohenbalken and Holloway.

The second approach is concerned with the application of conjugate gradient based methods to solve dual formulations of problems where the

objective function is separable, strictly convex and quadratic. These methods exploit both the sparsity and structure of the constraint matrix.

Conditions for finite and superlinear convergence of these techniques are also discussed, and computational results on a variety of large test problems are presented.

CHAPTER ONE

INTRODUCTION AND OVERVIEW

1.1 Introduction

This dissertation is concerned with the development of efficient algorithms for solving large-scale nonlinear optimization problems. The primary focus is on models with a pseudoconvex objective function and network constraints. These models have several important applications such as the standard traffic assignment problem [Beckmann 1951], electrical network problems [Cooper and Kennington 1977], water distribution models [Collins et al. 1978], economic and financial models [Bachem and Korte 1977, Mulvey 1985], computer and communication network models [Cantor and Gerla 1974], and many others. However, these algorithms are also applicable to problems with general linear constraints and, therefore, this research can be considered as applicable to a broad class of large-scale models.

The techniques considered in this study are divided in two classes: simplicial decomposition and dual methods.

Simplicial decomposition is a simple and direct method for dealing with large-scale pseudoconvex optimization problems, especially when the constraints are linear. The term "simplicial decomposition" is due to von Hohenbalken [1975, 1977], but the essential idea of this

decomposition has also been called "inner linearization/restriction" by Geoffrion [1970] and "the extended Frank-Wolfe method" by Holloway [1974]. The decomposition iterates by alternatively solving a linear programming subproblem that generates extreme points of the feasible region and a nonlinear master problem on the convex hull of a subset of previously generated extreme points (a simplex).

Thus, simplicial decomposition may be viewed as a form of modular nonlinear programming, provided that one has an effective computer code for solving the nonlinear master problem, as well as access to a code which can take advantage of the linearity of the subproblem, including any special structure which might be present. Recent developments in computational mathematical programming have provided such codes for many important applications, especially those which are modeled as network problems. For such models the linear subproblem can be solved by an appropriate network code such as GNET [Bradley et al. 1977], RNET [Grigoriadis and Hsu 1979], NETFLO [Kennington and Helgason 1980], or, in some instances, by a shortest path code [Syslo et al. 1983]. The master problem may be solved by a nonlinear code such as NPSOL [Gill et al. 1983], NLPQL [Schittkowski 1984], or the projected Newton method [Bertsekas 1982], all of which have the important feature of superlinear convergence. Without this feature the overall procedure might prove ineffective because too much time would be consumed for each master iteration.

The solution of a large-scale mathematical program can often be facilitated by exploiting its special structure. For instance, it is well-known that the dual of a strictly convex quadratic program subject

to linear equality constraints is an unconstrained quadratic program with as many variables as primal constraints [Wolfe 1961]. If, in addition, the primal problem is separable and has network constraints, the dual is very easy to find and does not require the storage of any matrix since the Hessian can be generated from node-arc network information. When the constraints are linear inequalities the dual is also a quadratic program but subject to simple nonnegativity constraints on the variables. For the special case of network problems, inequality constraints appear when the flow on the arcs is subject to upper and lower bounds.

The second part of this study discusses conjugate gradient based methods for solving dual formulations of separable strictly convex quadratic network problems. These methods are especially useful for large problems because they have minimum memory requirements, matrix operations can be simplified, and are finitely convergent under exact arithmetic.

1.2 Overview

Chapter Two gives a survey of the most well-known solution approaches for solving large-scale convex or pseudoconvex optimization problems with linear constraints that have been successfully applied to networks. This survey includes the Frank-Wolfe method [Frank and Wolfe 1956], the convex simplex method [Zangwill 1967], the reduced gradient method [Wolfe 1967], the piecewise-linear approximation method [Charnes and Lemke 1954], the primal truncated Newton method [Dembo 1983], and the relaxation method [Rockafellar 1984].

Chapter Three presents a restricted version of the simplicial decomposition algorithm in which the user can control the size of the master problem by a parameter, r , which limits the number of points defining the current simplex at each iteration. When $r=1$, the method is the familiar algorithm of Frank-Wolfe and, when r is greater than the number of variables, the method is the original simplicial decomposition.

Furthermore, we will obtain conditions under which the rules for dropping and retaining generated points guarantee that the method will terminate after a finite number of major iterations. Under these conditions, the overall convergence rate will be that of the algorithm selected for the master problem (e.g., superlinear). We also discuss a simpler formulation of the master problem and apply to it the projected Newton method of Bertsekas [1982] and a quasi-Newton version based on the BFGS formula [Dennis and More 1977], both of which have the property of superlinear convergence.

An extensive computational study of the basic restricted simplicial decomposition algorithm is also presented. The test problems consist of the linearly constrained Colville problems [Colville 1968], several traffic assignment models [Florian et al. 1983] which have become standard test problems, some simple, but large, electrical networks which were obtained from RCA physicists [Balberg et al. 1983], and the Dallas water distribution problem [Collins et al. 1978]. The experiments support the conclusion that when r can be chosen sufficiently large the decomposition rapidly achieves a solution to high accuracy. On problems where this is not possible, such as electrical

network problems, the experiments indicate that the approach will produce an objective value within 1 or 2 percent of the optimal in a relatively small amount of computer time.

The first part of Chapter Four specializes the conjugate gradient method of Hestenes and Stiefel [1952] to solve separable strictly convex quadratic network problems with undirected arcs. The network data structure is used to reduce memory requirements and computational effort in matrix operations. Computational experiments with electrical networks [Balberg et al. 1983] show that this approach is much superior to the primal method of Chapter Three.

At the end of Chapter Four, we deal with a dual ascent method for solving a Lagrangian relaxation of separable strictly convex quadratic networks with upper and lower bounds on the arcs. The ascent directions of this procedure are generated by conjugate gradient formulas (i.e., Fletcher and Reeves [1964], Polak and Ribiere [1969]).

CHAPTER TWO
PROBLEM STATEMENT AND PAST WORK

2.1 Problem Statement

The main concern of this dissertation is with the solution of the following nonlinear programming problem

$$\begin{array}{ll}\min & f(x) \\ \text{s.t.} & Ax = b \\ & l \leq x \leq u\end{array}\tag{2.1.1}$$

where $f(\bullet)$ is a continuously differentiable pseudoconvex function, $l \in (\mathbb{R} \cup \{\pm\infty\})^n$, $u \in (\mathbb{R} \cup \{\pm\infty\})^n$ and $b \in \mathbb{R}^m$ are constant vectors, $x \in \mathbb{R}^n$ is the vector of decision variables, and A is an $m \times n$ matrix. In many applications A will be the node-arc incidence matrix of a graph $G(V, E)$, where V is the set of nodes with cardinality m and E is the set of arcs with cardinality n .

2.2 Past Work

During the past decade a significant amount of research has been done in the development of new methods or extensions of existing techniques for solving large-scale nonlinear optimization problems of the form (2.1.1), with special emphasis on problems with network

constraints. Collins et al. [1978], Hanscom et al. [1978], Jacoby and Kowalik [1980], Dembo and Klincewicz [1981], Dembo [1983], Klincewicz [1983], and Kamesam and Meyer [1984] have studied the determination of steady-state flows and pressures in water distribution networks. Bachem and Korte [1977], Cottle and Duval [1982] and Kamesam and Meyer [1984] estimate data elements in input-output tables by solving quadratic transportation problems. Bertsekas [1976], Rosenthal [1981] and Kamesam and Meyer [1984] have studied the determination of optimal schedules in reservoir management problems. Determining the user equilibrium conditions in traffic networks has been studied by Dafermos and Sparrow [1969], Nguyen [1974, 1976], LeBlanc et al. [1975], Dembo and Tulowitzki [1983b], Lawphongpanich and Hearn [1983], Guelat [1983] and many others. These latter problems have multiple commodities.

Below, we summarize some of the most relevant solution approaches that have been used for (2.1.1).

2.2.1 Frank-Wolfe Method

The Frank-Wolfe (FW) method [Frank and Wolfe 1956] is the simplest and most straightforward method for solving pseudoconvex programming problems with linear constraints. Suppose that x^k is a feasible point at iteration k , and let y^k solve the following linearized subproblem

$$\begin{array}{ll} \min & \delta f(x^k)y \\ \text{s.t.} & Ay = b \\ & l \leq y \leq u \end{array}$$

where $\delta f(x^k)$ is the gradient of $f(\bullet)$ at x^k . Let α^k be a solution to the problem $\min \{f(\alpha x^k + (1 - \alpha)y^k) : 0 \leq \alpha \leq 1\}$, then the new iterate becomes $x^{k+1} = \alpha^k x^k + (1 - \alpha^k)y^k$. If $f(\bullet)$ is convex, $f(x^k) + \delta f(x^k)(y^k - x^k)$ is a lower bound for the optimal function value, and the algorithm terminates when this bound is within a certain tolerance.

The FW algorithm has appeal, especially for traffic assignment problems, since the subproblem becomes a set of independent shortest path problems.

Although the FW method progresses rapidly in the early iterations, it is known to converge rather slowly as the optimal solution is approached. Several authors, such as Collins et al. [1978], Florian et al. [1983] and LeBlanc et al. [1985], have improved its convergence by introducing a Partan direction and line search at each iteration.

2.2.2 Convex Simplex Method

The convex simplex (CS) method was introduced by Zangwill [1967] and may be viewed as a generalization of the linear simplex method. Let x^k denote a feasible point at iteration k with $l \leq x \leq u$ replaced by $0 \leq x \leq u'$. The linear equality constraints are partitioned as

$$A x = [B, N] \begin{bmatrix} x_B \\ x_N \end{bmatrix} = b$$

where x_B and x_N denote the basic and nonbasic variables with values of x_B^k and x_N^k , respectively. We also partition the gradient $\delta f(x^k)$ into basic and nonbasic components such that $\delta f(x^k) = [\delta f_B(x^k), \delta f_N(x^k)]$.

Let

$$r_N^k = \delta f_N(x^k) - N^T B^{-1} \delta f_B(x^k).$$

Then a nonbasic variable x_j^k is a candidate for value change if either $r_j^k < 0$ and $x_j^k < u_j^k$ or $r_j^k > 0$ and $x_j^k > 0$. If there is no candidate, then x^k is a solution. Otherwise, let x_j^k denote the candidate. The problem then becomes determining a step vector $\delta x^k = (\delta x_B^k, \delta x_N^k)$ which reduces the value of $f(x)$ while forcing $x^k + \delta x^k$ to satisfy the constraints. The only nonbasic variable that changes is x_j^k and δx_B^k will be changed so that the constraints remain satisfied. If δx_j^k is zero or the maximum allowable change, a pivot operation is performed where the leaving variable is any basic variable at zero or its upper bound. Otherwise, B^{-1} remains unchanged.

Collins et al. [1978] applied this method to network problems exploiting the underlying network structure using the same basic design as the linear network code NETFLO [Kennington and Helgason 1980].

2.2.3 Reduced Gradient Method

The reduced gradient (RG) method was first developed by Wolfe [1967] for linearly constrained problems and later generalized by Abadie and Carpenter [1969] to handle nonlinear constraints.

Here, we describe a version of the RG method for large-scale linearly constrained problems by Murtaugh and Saunders [1978] that has been the basis for more specialized methods for network problems. The linear equality constraints are partitioned as

$$A x = [B, S, N] \begin{bmatrix} x_B \\ x_S \\ x_N \end{bmatrix} = b$$

where the basic variables, x_B , are used to satisfy the equality constraints, the superbasic variables, x_S , are allowed to vary to minimize $f(x)$, and the nonbasic variables, x_N , are fixed at their bounds. B is a square nonsingular matrix. At each step, the problem then becomes determining a step vector $\delta x = (\delta x_B, \delta x_S, \delta x_N)$ which reduces the value of $f(x)$ while forcing $x + \delta x$ to satisfy the constraints. Since x_N is fixed, $\delta x_N = 0$, and allowing δx_S to be chosen freely, δx_B is determined by

$$\delta x_B = -B^{-1}S \delta x_S.$$

Considering the unconstrained problem of minimizing $f(x)$ as a function of the current set of superbasic variables, x_S , the reduced gradient h of $f(x)$ with respect to x_S is given by

$$h = \delta f_S(x) - S^T \pi$$

$$\pi = B^{-1} \delta f_B(x)$$

where $\delta f_B(x)$, $\delta f_S(x)$ and $\delta f_N(x)$ are the components of $\delta f(x)$ corresponding to x_B , x_S and x_N , respectively. Theoretically, the algorithm continues until $\|h\| = 0$, in which case $\delta x_S = 0$, and x_S is a solution in the manifold defined by x_B and x_S . The Lagrangian multipliers τ defined by

$$\tau = \delta f_N(x) - N^T \pi$$

are computed, and if $\tau_i \geq 0$ for each x_i in x_N at its lower bound and $\tau_i \leq 0$ for each x_i in x_N at its upper bound, x is a solution of (2.1.1). If not, a set of candidates can be chosen from x_N to become superbasic, and a new unconstrained problem is solved.

Murtaugh and Saunders use both quasi-Newton and conjugate gradient methods for minimization on the manifold.

Several authors have considered variants of the RG algorithm for network problems: Dembo and Klincewicz [1981] make explicit use of second-order information and exploit network programming data structure. Their algorithm includes a heuristic for conditioning the basis, a maximal basis procedure, and uses a scaled search direction. Beck et al. [1983] use a broad class of conjugate gradient methods to generate the search direction and make the algorithm useful for nonseparable objectives.

2.2.4 Piecewise-linear Approximation Method

The piecewise-linear approximation (PLA) method for separable convex programs was first introduced by Charnes and Lemke [1954]. Assume that the objective function of (2.1.1) is $f(x) = \sum_{i=1}^n f_i(x_i)$. This method simply chooses a set of grid points in the interval (l_i, u_i) , for $i = 1, \dots, n$, and replaces the functions $f_i(\bullet)$ by piecewise-linear approximations determined by the function values at the grid points. Then the approximating problem becomes a linear program. Suppose that K_i linear segments are to be used in the approximation of $f_i(\bullet)$. Then

the interval (l_i, u_i) is partitioned into K_i segments each of length u_{ik} , for $k = 1, \dots, K_i$, such that $\sum_{k=1}^{K_i} u_{ik} = u_i - l_i$. Let v_{ik} denote the end points of the K_i segments. That is

$$v_{ik} = l_i + \sum_{j=1}^k u_{ij}.$$

Let $x_i = l_i + \sum_{k=1}^{K_i} x_{ik}$ where x_{ik} denotes the value in segment k of (l_i, u_i) with unit cost of $c_{ik} = (f_i(v_{i,k+1}) - f_i(v_{ik}))/u_{ik}$. Then (2.1.1) can be approximated by

$$\min \sum_{i=1}^n f_i(l_i) + \sum_{i=1}^n \sum_{k=1}^{K_i} c_{ik} x_{ik}$$

$$\text{s.t. } Ax = b$$

$$0 \leq x_{ik} \leq u_{ik} \quad k = 1, \dots, K_i \\ i = 1, \dots, n.$$

Given that $\|df_i(x)\| \leq \alpha_i$, for $i = 1, \dots, n$, it is possible to obtain error bounds on the optimal function value depending on the constants α_i , the length of the maximum segment used in approximating the functions $f_i(\cdot)$ and the dual variables obtained by solving the linearized subproblem (see, e.g., Bazaraa and Shetty [1979]).

Several authors have used modifications of this technique for network problems: Collins et al. [1978] suggested a heuristic to find the unit costs c_{ik} based on the idea of least squares. Kamesam and Meyer [1984] discuss two algorithms. The first uses a local piecewise-linear approximation with a fixed number of segments (two, four or six segments per variable), and the second uses an implicit global approximation to the objective function that is modified at each

iteration. This second uses at most two piecewise-linear segments per variable at any time.

2.2.5 Primal Truncated Newton Method

The primal truncated Newton (PTN) method was introduced and specialized to nonlinear networks by Dembo [1983]. The PTN algorithm is a feasible direction method that fits into the convergent framework for large-scale optimization of Dembo and Sahi [1983]. This framework involves two types of feasible directions:

- i) Restricted directions: feasible descent directions restricted to lie in a subspace containing the current active set.
- ii) Relaxing directions: feasible descent directions along which one or more constraints may be relaxed.

The relaxing steps play an important role in the PTN algorithm. This role is to facilitate rapid identification of the optimal active set of constraints. For large problems it is crucial for the relaxing step to be able to make radical changes to the active set. One way to do so economically is to move in the direction of the negative gradient projected onto the box constrained problem (reduced problem with upper and lower bounds only).

For the restricted directions, PTN uses a primal feasible truncated Newton direction. Considering the same partition of the set of equality constraints of Subsection 2.2.3, define Z by

$$Z = \begin{bmatrix} -B^{-1}S \\ I \\ 0 \end{bmatrix} \quad \text{such that} \quad A Z = 0.$$

The restricted direction is then given by $p = Zp_s$, and the reduced gradient and Hessian are $Z^T \delta f(x)$ and $Z^T H(x) Z$, respectively, where $H(x)$ is the Hessian of $f(\bullet)$ at x . The descent direction p_s is computed by solving the reduced Newton equations

$$(Z^T H(x) Z) p_s = -Z^T \delta f(x)$$

inexactly using a linear conjugate gradient method. A forcing sequence is employed to reduce the inexactness as the method iterates. Accuracy is defined according to the relative residual $\|r\|/\|Z^T \delta f(x)\|$ in which $r = (Z^T H(x) Z) p_s + Z^T \delta f(x)$.

The global convergence of this method is based on the fact that the dropping of constraints is done according to a "forcing sequence strategy" and that the relaxing steps are "sufficiently long."

2.2.6 Relaxation Method

Relaxation (R) techniques draw from the field of duality theory and monotropic programming (see, e.g., Rockafellar [1984]). Apparently, Birkhoff and Diaz [1956] were the first to use relaxation techniques for convex networks. More recently, they have been used by Bertsekas and El Baz [1984] and Zenios and Mulvey [1986].

Consider the problem

$$\begin{aligned} \min \quad & \sum_{(i,j) \in E} f_{ij}(x_{ij}) \\ \text{s.t.} \quad & \sum_{(i,j) \in E} x_{ij} - \sum_{(j,i) \in E} x_{ji} = b_i \quad \text{for all } i \in V \\ & l_{ij} \leq x_{ij} \leq u_{ij} \quad \text{for all } (i,j) \in E \end{aligned}$$

where

$$f_{ij}(x_{ij}) = \begin{cases} \hat{f}_{ij}(x_{ij}) & \text{for } l_{ij} \leq x_{ij} \leq u_{ij} \\ \infty & \text{otherwise} \end{cases}$$

and $\hat{f}_{ij}(\bullet)$ is strictly convex.

The dual is the following unconstrained problem

$$\min \sum_{(i,j) \in E} f_{ij}^*(\pi_i - \pi_j)$$

where $f_{ij}^*(\bullet)$ is the convex conjugate function of $f_{ij}(\bullet)$, defined by

$$f_{ij}^*(t_{ij}) = \sup_{x_{ij}} \{t_{ij}x_{ij} - f_{ij}(x_{ij})\}$$

and π is defined as the price vector. The i^{th} price is really a Lagrangian multiplier associated with the i^{th} conservation of flow constraint.

Rockafellar [1970] gives necessary and sufficient conditions for optimality of the pair (x, π) , i.e., a feasible flow vector x and a dual vector π are optimal if and only if for all $(i,j) \in E$, $\pi_i - \pi_j$ is a subgradient of $f_{ij}(x_{ij})$ or, equivalently, $x_{ij} = \delta f_{ij}^*(\pi_i - \pi_j)$.

Since the dual problem is unconstrained and differentiable, the natural solution approaches are based on iterative descent procedures such as the Gauss-Siedel scheme for solving systems of equations. Optimization is achieved for one node at every iteration given information at adjacent nodes.

CHAPTER THREE

RESTRICTED SIMPLICIAL DECOMPOSITION

3.1 Background

The method described in this chapter is based on the standard simplicial decomposition (SD) algorithm first described by von Hohenbalken [1975, 1977] and Holloway [1974] for solving nonlinear programming problems with a pseudoconvex objective function and linear constraints. Conceptually, this method iterates by alternately solving a linear programming subproblem and a nonlinear master problem with simple convexity constraints. The subproblem generates extreme points of the feasible region, and the master problem finds the optimum of the objective function within the convex hull of a subset of previously generated points (a simplex).

The restricted version of the simplicial decomposition (RSD) algorithm was originally suggested by Lawphongpanich and Hearn [1983] with application to the traffic assignment problem. They showed that by retaining some old iterates together with some extreme points, the size of the master problem could be controlled by the user with a parameter r . Computational experiments indicated that the progress of the algorithm improved as r increased. The master problem was solved inexactly by a projection method on a quadratic approximation of the

objective function using a diagonal matrix. This method has a linear rate of convergence.

The RSD algorithm described in this chapter has two important new features:

- i) It will be shown that when the optimal solution is unique and r is larger than the dimension of the optimal face (see Wolfe [1970]) of the feasible region, RSD converges in a finite number of major cycles.
- ii) The master problem is solved to optimality using the projected Newton method of Bertsekas [1982] and a quasi-Newton version, both of which have a superlinear rate of convergence.

3.2 Preliminaries

As stated in Section 2.1, we are concerned with the following problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & Ax = b \\ & l \leq x \leq u \end{aligned} \tag{3.2.1}$$

where $l \in (\mathbb{R} \cup \{\pm\infty\})^n$, $u \in (\mathbb{R} \cup \{\pm\infty\})^n$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and A is an $m \times n$ matrix.

Consider the following assumptions:

Assumption 3.2.1: $f(\cdot)$ is a continuously differentiable pseudoconvex function.

Assumption 3.2.2: The feasible region of (3.2.1), denoted as S , is bounded.

Under Assumption 3.2.2 any element of S may be expressed as a convex combination of its extreme points, of which there are only a finite number. Thus, (3.2.1) can be rewritten in terms of the extreme points as

$$\begin{aligned} \min \quad & f\left(\sum_{i=1}^N \alpha_i a_i\right) \\ \text{s.t.} \quad & \sum_{i=1}^N \alpha_i = 1 \\ & \alpha_i \geq 0 \quad i = 1, \dots, N \end{aligned} \tag{3.2.2}$$

where N is the total number of extreme points and a_i corresponds to the i^{th} extreme point of S .

The following definition of a simplex from Rockafellar [1970] is important for the understanding of the algorithm.

Definition: Let $\{z_0, z_1, \dots, z_p\}$ be $p+1$ distinct points in R^n with $p \leq n$, where the vectors $z_1 - z_0, z_2 - z_0, \dots, z_p - z_0$ are linearly independent. Then, the set $C = H(z_0, z_1, \dots, z_p)$, the convex hull of $\{z_0, z_1, \dots, z_p\}$, is called a p -simplex in R^n . In addition, since C is always contained in a manifold of dimension p , a p -simplex is said to have dimension p , written as $\dim C = p$.

Caratheodory's Theorem (see, e.g., Bazaraa and Shetty [1979]) assures us that not only can every $x \in S$ be written as a convex combination of at most $n+1$ extreme points, but that x lies in the relative interior of at least one p -simplex defined by $p+1$ extreme points with $p \leq n$.

Now, after the following notation, we can state the original SD algorithm.

Notation:

- i) W_S = the set of retained extreme points.
- ii) W_X = a set which may be empty or contains one of the iterates.
- iii) $W = W_S \cup W_X$.
- iv) $|W|$ = the cardinality of W .
- v) $H(W)$ = the convex hull of W , i.e.,

$$H(W) = \{x : x = \sum \alpha_i a_i, \sum \alpha_i = 1, \alpha_i \geq 0 \text{ and } z_i \in W\}.$$
- vi) xy = the inner product of the vectors x and y .
- vii) $\|x\|$ = the euclidean norm of the vector x .

SD Algorithm

Step 0: Let x^0 be a feasible point of S . Set $W_S^0 = \emptyset$ and $k=0$.

Step 1: (Subproblem)

Let $y^k \in \arg \min \{\delta f(x^k)_y : y \in S\}$.

If $\delta f(x^k)(y^k - x^k) \geq 0$, x^k is a solution and terminate.

Otherwise, set $W_S^{k+1} = W_S^k \cup \{y^k\}$, and go to 2.

Step 2: (Master problem)

Let $x^{k+1} \in \arg \min \{f(x) : x \in H(W_S^{k+1})\}$.

Purge W_S^{k+1} of all the elements with zero weight in the expression of x^{k+1} as a convex combination of the elements of W_S^{k+1} . Set $k=k+1$ and go to 1.

3.3 Restricted Simplicial Decomposition and Global Convergence

In this section the basic version of RSD is stated and its convergence is proved. This basic algorithm differs from the one above in two respects:

i) The master problem feasible region here may be defined by a prior iterate as well as generated extreme points.

ii) When the number of retained extreme points equals a parameter r , the incoming extreme point replaces the extreme point with minimum weight in the expression of the current iterate as a convex combination of the retained extreme points and a prior iterate. This minimum weight "dropping rule" is given for definiteness and for the finiteness proof of the next section.

RSD Algorithm

Step 0: Let x^0 be a feasible point of S . Set $W_S^0 = \emptyset$, $W_x^0 = \{x^0\}$ and $k=0$.

Step 1: (Subproblem)

Let $y^k \in \arg \min \{\delta f(x^k)y : y \in S\}$.

If $\delta f(x^k)(y^k - x^k) \geq 0$, x^k is a solution and terminate.

Otherwise,

i) If $|W_S^k| < r$, set $W_S^{k+1} = W_S^k \cup \{y^k\}$ and $W_x^{k+1} = W_x^k$.

ii) If $|W_S^k| = r$, replace the element of W_S^k with the minimal weight in the expression of x^k as a convex combination of W^k with y^k to obtain W_S^{k+1} and let $W_x^{k+1} = \{x^k\}$.

Set $W^{k+1} = W_S^{k+1} \cup W_x^{k+1}$ and go to 2.

Step 2: (Master problem)

Let $x^{k+1} \in \arg \min \{f(x) : x \in H(W^{k+1})\}$.

Purge W_S^{k+1} and W_x^{k+1} of all the elements with zero weight in the expression of x^{k+1} as a convex combination of the elements of W^{k+1} . Set $k=k+1$ and go to 1.

As constructed, the feasible region for the master problem, $H(W^{k+1})$, always contains the current iterate, x^k , and the incoming extreme point, y^k . When x^k is not a solution, $y^k - x^k$ is a descent direction, thereby ensuring a decrease in the objective function value at the new iterate, x^{k+1} . The convergence proof then follows from the fact that x^{k+1} solves the master problem. The argument for the proof below is similar to that of Holloway [1974] and Lawphongpanich and Hearn [1983].

Lemma 3.3.1: $\delta f(x^*)(y - x^*) \geq 0$ for all $y \in S$ if and only if x^* is an optimal solution to (3.2.1).

Proof: This is a standard nonlinear programming result. ■

Lemma 3.3.2: If x^k is not optimal to (3.2.1), then $f(x^{k+1}) < f(x^k)$.

Proof: As noted earlier, x^k is feasible to the master problem. Since x^{k+1} solves the master problem

$$f(x^{k+1}) \leq f(x^k).$$

If $f(x^{k+1}) = f(x^k)$, then x^k is also a minimum and

$$\delta f(x^k)(y - x^k) \geq 0 \quad \text{for all } y \in H(W^{k+1})$$

but this is a contradiction since $y^k \in W^{k+1}$. ■

Lemma 3.3.3: Let $\{x^k\}$ be the sequence generated by RSD. Then, there cannot be a subsequence $\{x^k\}_K$ with the following three properties:

- i) $x^k \rightarrow x^\infty, k \in K$
- ii) $y^k \rightarrow y^\infty, k \in K$
- iii) $\delta f(x^\infty)(y^\infty - x^\infty) < 0$.

Proof: (By contradiction) Assume that such a subsequence, K , exists, i.e., there exists a $r > 0$ such that

$$\delta f(x^\infty)(y^\infty - x^\infty) < -r.$$

Since $f(\bullet)$ is continuously differentiable, there exists a $\pi \in K$ sufficiently large so that for any $k \geq \pi$

$$\delta f(x^k)(y^k - x^k) < -r/2$$

and a $\phi > 0$ such that for $0 \leq t \leq \phi$

$$[\delta f(x^k + t(y^k - x^k))](y^k - x^k) < -r/4$$

By construction, both y^k and x^k are feasible to the master problem at iteration k . Hence, there exists $t^* \in (0, 1)$ such that $x^k + t^*(y^k - x^k)$ is feasible to the master problem. By the optimality of x^{k+1}

$$f(x^{k+1}) \leq f(x^k + t^*(y^k - x^k))$$

and by Taylor's expansion

$$f(x^{k+1}) \leq f(x^k) + \delta f(x^k) + \alpha t^*(y^k - x^k))t^*(y^k - x^k)$$

where $0 \leq \alpha \leq 1$. Since $0 \leq \alpha t^* \leq \phi$

$$f(x^{k+1}) \leq f(x^k) - t^*(r/4). \quad (3.3.1)$$

Because $f(\bullet)$ is continuous and $\{f(x^k)\}$ is monotonically decreasing (see Lemma 3.3.2), the limit of $\{f(x^k)\}$, $f(x^\infty)$, exists. For k sufficiently large

$$f(x^\infty) \geq f(x^k) - t^*(r/8). \quad (3.3.2)$$

For $k \geq \pi$ and sufficiently large so that (3.3.2) holds, (3.3.1) and (3.3.2) imply that

$$f(x^\infty) \geq f(x^k) - t^*(r/8) \geq f(x^k) - t^*(r/4) \geq f(x^{k+1}) > f(x^k).$$

Since $\{f(x^k)\}$ decreases monotonically, this is a contradiction. Thus, the lemma is proved. \square

Theorem 3.3.1: Given that $f(\bullet)$ is continuously differentiable, RSD either terminates at a solution or generates a sequence $\{x^k\}$ for which any subsequential limit is a solution.

Proof: If RSD terminates, the current iterate x^k must satisfy the stopping criterion in Step 1. By Lemma 3.3.1, x^k must be a solution. When a sequence $\{x^k\}$ is generated, Lemma 3.3.3 guarantees that the limit of every convergent subsequence is a solution. \square

In addition, when (3.2.1) has a unique solution, e.g., when $f(\bullet)$ is strictly pseudoconvex, the entire sequence $\{x^k\}$ converges to the optimal solution (see, e.g., Bazaraa and Shetty [1979]).

3.4 Finite Convergence

The von Hohenbalken [1975, 1977] proof of finiteness for the standard SD relies on the fact that all positively weighted extreme points are retained from one iteration to the next. His proof also utilizes Caratheodory's Theorem which implies that at most $n+1$ extreme points define each iterate. For these reasons, the proof does not apply to RSD.

To prove finiteness of RSD, the following properties of simplices from Rockafellar [1970] are necessary:

Property 1: If x is an element of a p -simplex, C , then x can be uniquely expressed as a convex combination of the points z_0, z_1, \dots, z_p defining C , i.e.,

$$x = \sum_{i=0}^p \alpha_i z_i$$

$$\sum_{i=0}^p \alpha_i = 1$$

$$\alpha_i \geq 0 \quad i = 0, 1, \dots, p$$

and $\alpha_0, \alpha_1, \dots, \alpha_p$ are unique.

Property 2: If x is an element of a p -simplex, C , and the convex weight, α_i , for some $i = 0, 1, \dots, p$, is positive in the (unique) expression of x as a convex combination of z_0, z_1, \dots, z_p , then $H(z_0, z_1, \dots, z_{i-1}, x, z_{i+1}, \dots, z_p)$ is also a p -simplex.

Property 3: If $H(z_0, z_1, \dots, z_p)$ is a p -simplex, then $H(z_0, z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_p)$, for some $i = 0, 1, \dots, p$, is a $(p-1)$ -simplex.

We need the following additional assumption:

Assumption 3.4.1: (3.2.1) has a unique solution, denoted as x^* , i.e., $f(\bullet)$ is strictly pseudoconvex.

Under the above assumption we can define the following set:

$$I^* = \{a_i : \delta f(x^*)(a_i - x^*) = 0, i = 1, 2, \dots, N\}.$$

We refer to $H(I^*)$ as the optimal face (see Wolfe [1970]) of S . Note that $\delta f(x^*)(a_i - x^*) > 0$ for all $a_i \notin I^*$.

The following two lemmas are necessary for the finiteness of RSD.

Lemma 3.4.1: Given that $x'' \in \arg \min \{f(x) : x \in H(z_1, z_2, \dots, z_p)\}$

where each z_i is an arbitrary element of R^n , and x'' is written as

$\sum_{i=1}^p \alpha_i z_i$, where $\sum_{i=1}^p \alpha_i = 1$, and $\alpha_i \geq 0, i = 1, \dots, p$, then

$\delta f(x'')(z_j - x'') > 0$ implies that $\alpha_j = 0$.

Proof: (By contradiction) Assume that $\alpha_j > 0$. Define

$$\begin{aligned} z &= x'' + \alpha_j(x'' - z_j)/(1 - \alpha_j) \\ &= x''/(1 - \alpha_j) - \alpha_j z_j/(1 - \alpha_j) \\ &= \sum_{i=1, i \neq j}^p \alpha_i z_i / (1 - \alpha_j) \end{aligned}$$

and

$$\sum_{i=1, i \neq j}^p \alpha_i / (1 - \alpha_j) = (1 - \alpha_j) / (1 - \alpha_j) = 1.$$

Therefore, z is an element of $H(z_1, z_2, \dots, z_p)$ and $d = (z - x'')$ is a feasible direction. However

$$\begin{aligned}\delta f(x'')d &= \delta f(x'')(z - x'') = \{\alpha_j \delta f(x'')(x'' - z_j)\} / (1 - \alpha_j) \\ &= \{-\alpha_j \delta f(x'')(z_j - x'')\} / (1 - \alpha_j) \\ &< 0\end{aligned}$$

i.e., $d = z - x''$ is a descent direction, a contradiction to the optimality of x'' . Therefore, α_j must be zero. ■

Lemma 3.4.2: Let $\{x^k\}$ be an infinite sequence converging to x^* , the optimal solution to the problem : $\min \{f(x) : x \in S\}$, where $S = H(a_1, a_2, \dots, a_N)$. If $f(\bullet)$ is continuously differentiable, then there exists an integer, π , such that for any $k > \pi$ the following hold:

- i) $\delta f(x^k)(a_j - x^k) > 0$ for all $a_j \notin I^*$.
- ii) $\min \{\delta f(x^k)(y - x^k) : y \in S\}$
 $= \min \{\delta f(x^k)(y - x^k) : y \in I^*\}.$

Proof: For each $a_j \notin I^*$, the continuity of $\delta f(\bullet)$ implies that

$$\lim_{k \rightarrow \infty} \delta f(x^k)(a_j - x^k) = \delta f(x^*)(a_j - x^*) = r_j$$

for some $r_j > 0$. For any σ such that $0 < \sigma < \min \{r_j : a_j \notin I^*\}$ there must exist a positive integer, τ_j , such that for any $k > \tau_j$

$$\begin{aligned}|\delta f(x^k)(a_j - x^k) - \delta f(x^*)(a_j - x^*)| &< (r_j - \sigma) \\ -(r_j - \sigma) &< \delta f(x^k)(a_j - x^k) - r_j < (r_j - \sigma) \\ \sigma &< \delta f(x^k)(a_j - x^k) < 2r_j - \sigma.\end{aligned}$$

Then, for any σ between 0 and $r' = \min \{r_j : a_j \notin I^*\}$, the following holds for any $k > \pi = \max \{\tau_j : a_j \notin I^*\}$

$$\delta f(x^k)(a_j - x^k) > \sigma \quad \text{for all } a_j \notin I^*.$$

Since $\sigma > 0$, this proves (i).

Note that when x^k is not optimal, $\delta f(x^k)(y^k - x^k)$ must be negative. From (i), any $a_j \notin I^*$ cannot be a solution to the subproblem. That is

$$\begin{aligned} \min \{ \delta f(x^k)(y - x^k) : y \in S \} \\ = \min \{ \delta f(x^k)(y - x^k) : y \in I^* \}. \end{aligned}$$

The following theorem describes the behavior of the algorithm with respect to the relation between W_S^k and I^* , and holds for all $r \geq 1$.

Theorem 3.4.1: If RSD generates an infinite sequence $\{x^k\}$ converging to x^* , then there exists a positive integer, π , such that for any $k > \pi$ the following hold:

- i) The incoming extreme point at iteration k , y^k , belongs to I^* .
- ii) W_S^k is a subset of I^* .

Proof: Let π be as defined in Lemma 3.4.2, i.e., $\pi = \max \{ \tau_j : a_j \notin I^* \}$. Then, (i) follows directly from Lemma 3.4.2(ii).

To prove (ii), assume that for some $k > \pi$, W_S^k is not a subset of I^* . Without loss of generality, suppose that the elements a_1, a_2, \dots, a_q from W_S^k do not belong to I^* . Then, Lemma 3.4.2(i) implies that at the end of Step 2 and for $j = 1, 2, \dots, q$

$$\delta f(x^{k+1})(a_j - x^{k+1}) > 0$$

which, by Lemma 3.4.1, further implies that the weight of a_j , for $j = 1, 2, \dots, q$, in the expression of x^{k+1} as a convex combination of W^{k+1} must be all zero. Therefore, we have that a_1, a_2, \dots, a_q satisfy the column dropping criteria in Step 2, which means that the set W_S^{k+1} will consist entirely of elements from the set I^* . Moreover, (i) ensures

that only elements from the set I^* will be added to the set W_S^k in subsequent iterations, and the desired result is obtained by letting π equal $k+1$. ■

The result of the lemma below implies that the union of W^k and $\{y^k\}$, the solution to the subproblem, always forms a simplex. This leads to the result of Theorem 3.4.2 which states that, for $k \geq 0$, the set W^k always forms a simplex. Then, it is shown in Theorem 3.4.3 that, when r is larger than the dimension of $H(I^*)$, the algorithm must stop after having generated a finite number of simplices.

Lemma 3.4.3: Let $C = H(z_0, z_1, \dots, z_p)$ be a p -simplex which is also a subset of a convex feasible region S . Furthermore, let $x' \in \arg \min \{f(x) : x \in C\}$ which satisfies

$$x' = \sum_{i=0}^p \beta_i z_i$$

$$\sum_{i=0}^p \beta_i = 1$$

$$\beta_i > 0 \quad i = 0, 1, \dots, p.$$

If x' does not minimize $f(x)$ over S , then there exists a $y \in S$ such that $\delta f(x')(y - x') < 0$ and $H(z_0, z_1, \dots, z_p, y)$ is a $(p+1)$ -simplex.

Proof: (By contradiction) Assume that $H(z_0, z_1, \dots, z_p, y)$ is not a $(p+1)$ -simplex, i.e., the vectors: $z_1 - z_0, \dots, z_p - z_0, y - z_0$, are linearly dependent. In other words, there exist coefficients α_i , for $i = 1, \dots, p$, satisfying

$$y - z_0 = \sum_{i=1}^p \alpha_i (z_i - z_0)$$

$$y = (1 - \sum_{i=1}^p \alpha_i)z_0 + \sum_{i=1}^p \alpha_i z_i$$

$$y = \sum_{i=0}^p \alpha_i z_i \quad \text{where } \alpha_0 = 1 - \sum_{i=1}^p \alpha_i.$$

Moreover, it follows from the optimality of x' over C that

$$\delta f(x')(z_i - x') \geq 0 \quad i = 0, 1, \dots, p$$

and, from Lemma 3.4.1, this implies that

$$\delta f(x')(z_i - x') = 0 \quad i = 0, 1, \dots, p$$

since $\beta_i > 0$ for all i . However

$$\begin{aligned} 0 &> \delta f(x')(y - x') = \delta f(x')\left(\left(\sum_{i=0}^p \alpha_i z_i\right) - x'\right) \\ &= \sum_{i=0}^p \alpha_i [\delta f(x')(z_i - x')] = 0. \end{aligned}$$

This is a contradiction and $H(z_0, z_1, \dots, z_p, y)$ must be a $(p+1)$ -simplex. \square

Theorem 3.4.2: In RSD, the set $H(W^k)$ is a simplex for all $k \geq 0$.

Proof: We show by induction that $H(W^k)$ is a simplex at the start of Step 1. It then follows that $H(W^k)$ is a simplex at every step of RSD.

When $k = 0$, $W_x^0 = \{x^0\}$ and $W_s^0 = \emptyset$. Therefore, $W^0 = \{x^0\}$ and $H(W^0)$ is a 0-simplex. Assume that $H(W^k)$ is a p -simplex for $k \geq 0$. Then, it must be shown that $H(W^{k+1})$ is also a simplex.

Assume, without loss of generality, that $W_s^k = \{a_0, a_1, \dots, a_{p-1}\}$ and $W_x^k = \{x'\}$, and by assumption $W^k = W_s^k \cup W_x^k$ defines a p -simplex. The

elements with zero convex weight have been discarded at the end of the preceding Step 2. Therefore, the remaining elements in W^k at the beginning of Step 1 must have positive weights. Since $\delta f(x^k)(y^k - x^k) \geq 0$ implies that the algorithm must stop and W^{k+1} is not generated, it is assumed below that $\delta f(x^k)(y^k - x^k) < 0$, and either Step 1(i) or 1(ii) will be executed depending on the cardinality of W^k .

In Step 1(i), $W^{k+1} = W^k \cup \{y^k\}$, and $H(W^{k+1})$ defines a $(p+1)$ -simplex because of Lemma 3.4.3.

In Step 1(ii), $W^{k+1} = \{a_0, a_1, \dots, a_{i-1}, y^k, a_{i+1}, \dots, a_{p-1}, x^k\}$, where a_i has the minimum weight. In this case, W^{k+1} defines a p -simplex because $H(a_0, a_1, \dots, a_{p-1}, x^k, y^k)$ is a $(p+1)$ -simplex by Lemma 3.4.3, $H(a_0, a_1, \dots, a_{p-1}, x^k, y^k)$ is a $(p+1)$ -simplex by Property 2, and $H(a_0, a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_{p-1}, x^k, y^k)$ is a p -simplex by Property 3. Thus, in either case, $H(W^{k+1})$ is a simplex.

At the end of Step 2, all zero weighted elements of W^{k+1} are discarded and Property 3 guarantees that the convex hull of the remaining elements still forms a simplex. Since the set W^{k+1} at the end of Step 2 is the same as the one at the beginning of Step 1, the theorem is proved. ■

Theorem 3.4.3: When x^* is unique and $r \geq \dim I^* + 1$, RSD converges to x^* after a finite number of executions of Step 1.

Proof: (By contradiction) Assume that the algorithm generates an infinite sequence $\{x^k\}$ and, because of the uniqueness of the solution, $\{x^k\}$ must converge to x^* . For every k , $H(W^k)$ is a simplex by Theorem 3.4.2, and for $k > \pi$, W_S^k is a subset of I^* , where π is as defined in Theorem 3.4.1. Since $H(W^k)$ is a simplex and $W^k = W_S^k \cup W_x^k$, W_S^k must also

define a simplex by Property 3. Below, it is shown that, at the beginning of Step 1, $|W_S^k| < r$ for all $k > \pi$.

Since the algorithm does not stop at any iteration $k > \pi$, $\delta f(x^k)(y^k - x^k) < 0$ and $H(W^k \cup \{y^k\})$ is a simplex by Lemma 3.4.3. By Theorem 3.4.1(i), $y^k \in I^*$. Thus, $W_S^k \cup \{y^k\}$ is still a subset of I^* , and

$$\begin{aligned} |W_S^k| - 1 &= \dim W_S^k < \dim (W_S^k \cup \{y^k\}) \leq \dim I^* \\ |W_S^k| &< \dim I^* + 1 \leq r \end{aligned}$$

This last inequality implies that Step 1(ii) is never executed after the π^{th} iteration, and W^k is a subset of $I^* \cup W_X^\pi$, for all $k > \pi$. Moreover, the objective function values at the iterates x^k strictly decrease after each iteration, thereby implying that the set W^k cannot be repeated in the sequence $\{W^k\}_{k>\pi}$. This is impossible, since there are only a finite number of distinct subsets of $I^* \cup W_X^\pi$ with size $r+1$ or less. So, the algorithm must terminate after a finite number of executions of Step 1. ■

3.5 Master Problem

The master problem of (3.2.1) may be formulated as follows

$$\begin{aligned} \min \quad & f(A^k w) \\ \text{s.t.} \quad & \sum_{i=1}^s w_i = 1 \\ & w_i \geq 0 \quad i = 1, \dots, s \end{aligned} \tag{3.5.1}$$

where $A^k = [a_1, a_2, \dots, a_s]$ is the matrix whose columns are the elements of w^k , $w = [w_1, w_2, \dots, w_s]^T$ are the associated weights, and $s \leq r + 1$.

Since the master problem is solved to optimality and Lemma 3.3.2 shows that after each iteration of RSD the objective function value decreases, the weight corresponding to the last generated extreme point must be positive. The theorem below uses this fact to eliminate the equality constraint from (3.5.1) and reduce its dimension by 1.

Theorem 3.5.1: Given that $w_s^* > 0$, where a_s is the last generated extreme point, and $f(\bullet)$ is pseudoconvex, (3.5.1) is equivalent to

$$\begin{aligned} \min \quad & h(z) = f(B^k z + a_s) \\ \text{s.t.} \quad & z_i \geq 0 \quad i = 1, \dots, s-1 \end{aligned} \quad (3.5.2)$$

where $B^k = [a_1 - a_s, a_2 - a_s, \dots, a_{s-1} - a_s]$.

Proof: The Karush-Kuhn-Tucker (KKT) conditions of (3.5.1) are

$$\delta_i f(A^k w^*) + \mu + \tau_i = 0 \quad i = 1, \dots, s \quad (3.5.3)$$

$$w_i^* \tau_i = 0 \quad i = 1, \dots, s \quad (3.5.4)$$

$$\tau_i \geq 0 \quad i = 1, \dots, s \quad (3.5.5)$$

and w^* feasible.

$w_s^* > 0$ and $w_s^* \tau_s = 0$ imply that $\tau_s = 0$. In addition, $\tau_s = 0$ and $\delta_s f(A^k w^*) + \mu + \tau_s = 0$ imply that $\mu = -\delta_s f(A^k w^*)$.

The KKT conditions of (3.5.2) are

$$\delta_i h(z^*) + \tau_i^* = 0 \quad i = 1, \dots, s-1 \quad (3.5.6)$$

$$z_i^* \tau_i^* = 0 \quad i = 1, \dots, s-1 \quad (3.5.7)$$

$$\tau_i^* \geq 0 \quad i = 1, \dots, s-1 \quad (3.5.8)$$

and z^* feasible.

Since $\delta_i h(z^*) = \delta_i f(A^k w^*) - \delta_s f(A^k w^*)$ for $i = 1, \dots, s-1$, (3.5.6) may be rewritten as

$$\delta_i f(A^k w^*) - \delta_s f(A^k w^*) + \tau_i^* = 0 \quad i = 1, \dots, s-1 \quad (3.5.9)$$

$$\text{or } \delta_i f(A^k w^*) + \mu + \tau_i^* = 0 \quad i = 1, \dots, s-1. \quad (3.5.10)$$

Note that (3.5.7), (3.5.8) and (3.5.10) are also the KKT conditions for (3.5.1). Thus, if w^* is an optimal solution of (3.5.1), then $[w_1^*, \dots, w_{s-1}^*]^T$ will be an optimal solution of (3.5.2).

Since the master problem must be solved repeatedly, it is important that a superlinearly convergent method be used, and this requires the storage of an adequate approximation of the Hessian matrix. However, this matrix is at most rxr in RSD. Further, the constraints are simple, making the projected Newton method of Bertsekas or its quasi-Newton versions a natural choice.

3.5.1 Bertsekas Projection Method [Bertsekas 1982]

The projection method of Bertsekas for (3.5.2) is an iterative procedure of the following form

$$z^{k+1} = [z^k - \alpha^k D^k \delta h(z^k)]^+ \quad (3.5.11)$$

where $[\cdot]^+$ denotes projection on the positive orthant, α^k is a stepsize chosen by an Armijo-like rule and D^k is a positive definite symmetric matrix which is partly diagonal. The matrix D^k can be computed on the

basis of first or second derivatives of $h(\bullet)$ so that the resulting method becomes a quasi-Newton or a Newton algorithm.

The following two assumptions are necessary for the convergence of the sequence $\{z^k\}$ to a critical point:

Assumption 3.5.1: The gradient $\delta h(\bullet)$ is Lipschitz continuous on each bounded set of \mathbb{R}^n , i.e., given any bounded set S in \mathbb{R}^n , there exists a scalar L (depending on S) such that

$$\|\delta h(z_1) - \delta h(z_2)\| \leq L \|z_1 - z_2\| \quad \text{for any } z_1, z_2 \in S.$$

Assumption 3.5.2: There exists positive scalars τ_1, τ_2 and nonnegative integers q_1, q_2 such that

$$\tau_1 (w^k)^{q_1} \|x\|^2 \leq x^D x \leq \tau_2 (w^k)^{q_2} \|x\|^2 \quad \text{for all } x \in \mathbb{R}^n, k = 0, 1, \dots$$

where $w^k = \|z^k - [z^k - M\delta h(z^k)]^+\|$ and M is a positive definite diagonal matrix.

The algorithm described below utilizes a scalar $\epsilon > 0$ (typically small), a fixed diagonal matrix M (for example the identity), and two parameters $\beta \in (0, 1)$ and $\sigma \in (0, \frac{1}{2})$ that will be used in connection with an Armijo-like stepsize rule.

Step 0: Let z^0 be a feasible point. Set $k=0$.

Step 1: Select a positive definite matrix D^k which is diagonal with respect to the set I_+^k given by

$$I_+^k = \{i : 0 \leq z_i^k \leq \epsilon^k, \delta_i h(z^k) > 0, i = 1, \dots, s\}$$

where $\epsilon^k = \min \{\epsilon, w^k\}$ and $w^k = \|z^k - [z^k - M\delta h(z^k)]^+\|$.

Go to 2.

Step 2: Denote $p^k = D^k \delta h(z^k)$

and $z^k(\alpha) = [z^k - \alpha p^k]^+$ for all $\alpha \geq 0$.

Find z^{k+1} given by

$$z^{k+1} = z^k(\alpha^k)$$

where $\alpha^k = \beta^m$ and m is the first nonnegative integer such that

$$h(z^k) - h(z^k(\beta^m)) \geq$$

$$\sigma \left\{ \beta^m \sum_{i \in I_+^k} \delta_i h(z^k) p^k + \sum_{i \in I_+^k} \delta_i h(z^k) [z^k - z^k(\beta^m)] \right\}.$$

If $h(z^{k+1}) = h(z^k)$, z^{k+1} is a solution and terminate.

Otherwise, set $k=k+1$ and go to 1.

The stepsize rule may be viewed as a combination of the Armijo-like rule [Bertsekas 1976] and the Armijo rule usually employed in unconstrained minimization (see, e.g., Luenberger [1984]).

The following assumption guarantees that if the algorithm generates a sequence $\{z^k\}$ with a limit point z^* , then the set of active constraints at z^* is identified in a finite number of iterations and the algorithm is equivalent to an unconstrained optimization method restricted on T , the subspace of binding constraints at z^* .

Assumption 3.5.3: The local minimum z^* of $\min \{h(z) : z \geq 0\}$ is such that for some $\delta > 0$, $h(\bullet)$ is twice continuously differentiable in the open sphere $\{z : \|z - z^*\| < \delta\}$, and there exist positive scalars m_1, m_2 such that

$$m_1 \|x\|^2 \leq x \delta^2 h(z) x \leq m_2 \|x\|^2$$

for all z such that $\|z - z^*\| < \delta$, and $x \neq 0$ such that $x_i = 0$ for all $i \in B(z^*)$, where $B(z) = \{i : z_i = 0\}$. Furthermore,

$$\delta_i h(z^*) > 0 \quad \text{for all } i \in B(z^*).$$

By Assumption 3.5.3, $\delta^2 h(z^*)$ is positive definite on T for k sufficiently large. This implies that if the portion of the matrix D^k corresponding to the indices $i \notin I_+^k$ is chosen to be the inverse of the Hessian or an approximation of the inverse of the Hessian of $h(\bullet)$ with respect to these indices then the algorithm eventually reduces to a Newton or quasi-Newton method restricted on the subspace T . This type of argument can be used to construct a number of Newton or quasi-Newton methods and prove corresponding convergence and rate of convergence results. The following are two forms of the matrix D^k that have been used in our computational experiments:

i) Newton Method:

$$D^k = (H^k)^{-1}$$

where H^k is the matrix with elements H_{ij}^k given by

$$H_{ij}^k = \begin{cases} 0 & \text{if } i \neq j \text{ and either } i \in I_+^k \text{ or } j \in I_+^k \\ \delta_{ij}^2 h(z^k) & \text{otherwise.} \end{cases}$$

ii) Quasi-Newton Method:

$$D^k = (Q^k)^{-1}$$

where Q^k is the matrix with elements Q_{ij}^k given by the BFGS formula [Dennis and More 1977]. We first recall the BFGS formula:

$$B^{k+1} = B^k + \frac{y^k y^{kT}}{y^k y^k} - \frac{B^k s^k (B^k s^k)^T}{s^k B^k s^k}$$

where $s^k = z^{k+1} - z^k$

and $y^k = \delta h(z^{k+1}) - \delta h(z^k)$.

Define $\phi^k = y^k s^k$. It is known [Powell 1978] that if B^k is positive definite, so is B^{k+1} if and only if $\phi^k > 0$. To preserve the positive definiteness of B^k , if $\phi^k \leq 0$, we generalize the BFGS formula as in Powell:

$$B^{k+1} = B^k + \frac{\tau^k \tau^{kT}}{\tau^k s^k} - \frac{B^k s^k (B^k s^k)^T}{s^k B^k s^k}$$

with

$$\phi = \frac{0.8 s^k B^k s^k}{s^k B^k s^k - y^k s^k}$$

$$\tau^k = \begin{cases} \phi y^k + (1 - \phi) B^k s^k & \text{if } \phi^k \leq 0.2 s^k B^k s^k \\ y^k & \text{otherwise.} \end{cases}$$

Now, we define Q^k by

$$Q_{ij}^k = \begin{cases} 0 & \text{if } i \neq j \text{ and either } i \in I_+^k \text{ or } j \in I_+^k \\ B_{ij}^k & \text{otherwise.} \end{cases}$$

3.6 Implementation

A broad class of real world problems have been solved to test the basic RSD algorithm. The test problems consist of the linearly constrained Colville problems [Colville 1968] used by von Hohenbalken [1977] to test his APL code for SD, several traffic assignment models which range up to 2836 arcs and which have become standard test problems, the 667 node Dallas water distribution problem [Collins et al. 1978], and some simple, but large, electrical networks which were obtained from RCA physicists studying the properties of conducting fibers imbedded in an insulating polymer [Balberg et al. 1983]. The test problems required different codes for the linear subproblems, according to the type of model. For the Colville problems, a local LP code was developed. This code allows starting each subproblem, after the first, from the optimal solution of the prior iterate. This is an obviously important feature for efficiency of the decomposition. For the traffic assignment problems, the subproblems become shortest path problems [Nguyen 1976] and a code of Dijkstra's method by Nguyen and James [1975] was employed. Finally, the convex single commodity minimum cost flow problems have as subproblems their linear version for which there exist many algorithms. For the experiments below, the primal simplex code NETFLO [Kennington and Helgason 1980], modified to accommodate real variables and to allow starting from the prior solution, was chosen. In some instances, e.g., the electrical network models, the subproblem may also be solved by a shortest path code.

Both the water distribution and electrical network problems have unconstrained variables in their formulations and this implies that the subproblem can be unbounded. However, the solution for each problem is known to lie in the convex hull of some extreme points. To illustrate these cases, consider the problem

$$\begin{aligned} \min \quad & \sum_{i=1}^n f_i(x_i) \\ \text{s.t.} \quad & Ax = b \end{aligned} \quad (3.6.1)$$

where $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and A is an $m \times n$ node-arc incidence matrix.

Consider the following assumption:

Assumption 3.6.1: $f_i(\cdot)$ is strictly convex, differentiable and symmetric about zero, i.e., $f_i(x_i) = f_i(-x_i)$ for $i = 1, 2, \dots, n$.

The subproblem of (3.6.1) in RSD is defined by

$$\begin{aligned} \min \quad & \sum_{i=1}^n \delta f_i(x_i) y_i \\ \text{s.t.} \quad & Ay = b. \end{aligned} \quad (3.6.2)$$

Note that (3.6.2) may be unbounded due to the existence of negative cycles which may render some inefficiency in RSD. Although the problem with undirected arcs can be remedied by replacing each undirected arc with two directed arcs, the problem of negative cycles still exists. An alternative solution is to place artificially large upper bounds on all arcs which could lead to the following undesirable properties: i) slow progress in the first few iterations, and ii) artificially low lower bounds on the optimal objective function value.

An alternative to solve these problems is to replace the standard Subproblem (3.6.2) by a new one that always generates an extreme point and produces descent directions. The following theorem gives two problems equivalent to (3.6.1), the subproblems of which may be used instead of (3.6.2). By equivalent we mean that an optimal solution in one induces an optimal solution for the others.

Theorem 3.6.1: Under Assumption 3.6.1, the Problem (3.6.1) is equivalent to the following two:

$$\begin{aligned} \min \quad & \sum_{i=1}^n f_i(x_i^+ + x_i^-) \\ \text{s.t.} \quad & A(x^+ - x^-) = b \\ & x_i^+, x_i^- \geq 0 \quad i = 1, \dots, n \end{aligned} \tag{3.6.3}$$

and

$$\begin{aligned} \min \quad & \sum_{i=1}^n [f_i(x_i^+) + f_i(x_i^-) - f_i(0)] \\ \text{s.t.} \quad & A(x^+ - x^-) = b \\ & x_i^+, x_i^- \geq 0 \quad i = 1, \dots, n. \end{aligned} \tag{3.6.4}$$

Proof: We need to show that (3.6.1) is equivalent to (3.6.3) and to (3.6.4). However, in the interest of brevity, we will prove only the first equivalency, since the second can be done in a similar manner.

Let (x^+, x^-) be an optimal solution to (3.6.3). Then, we establish the following two properties:

- i) $x_i^+ x_i^- = 0$ for $i = 1, \dots, n$.
- ii) $x = x^+ - x^-$ is an optimal solution to (3.6.1).

To prove (i), assume the converse, i.e., there exists an index j such that $x_j^+ > 0$ and $x_j^- > 0$. Define

$$\begin{aligned} z_j^+ &= \max \{0, x_j^+ - x_j^-\} \\ z_j^- &= \max \{0, x_j^- - x_j^+\} \end{aligned}$$

and let

$$z_i^+ = x_i^+ \text{ and } z_i^- = x_i^- \quad \text{for all } i \neq j.$$

Then, by definition of (z^+, z^-)

$$z_j^+ < x_j^+, \quad z_j^- < x_j^- \quad \text{and} \quad z_j^+ z_j^- = 0.$$

Moreover, due to the strict convexity and symmetry of $f_i(\bullet)$

$$f_j(z_j^+ + z_j^-) < f_j(x_j^+ + x_j^-)$$

$$\text{and } f_i(z_i^+ + z_i^-) = f_i(x_i^+ + x_i^-) \quad \text{for all } i \neq j.$$

That is, (z^+, z^-) is a better solution which satisfies property (i), and we have a contradiction.

To prove (ii), assume the converse, i.e., x is not an optimal solution to (3.6.1). Let z be an optimal solution to (3.6.1). Then

$$\sum_{i=1}^n f_i(z_i) < \sum_{i=1}^n f_i(x_i)$$

Define

$$z_i^+ = \max \{0, z_i\} \text{ and } z_i^- = \max \{0, -z_i\} \quad \text{for } i = 1, \dots, n.$$

Then, (z^+, z^-) is feasible to (3.6.3) and

$$\sum_{i=1}^n f_i(z_i^+ + z_i^-) = \sum_{i=1}^n f_i(z_i) < \sum_{i=1}^n f_i(x_i) = \sum_{i=1}^n f_i(x_i^+ + x_i^-).$$

But this means that (x^+, x^-) cannot be optimal to (3.6.3) which is a contradiction.

Let x be an optimal solution to (3.6.1). Define

$$x_i^+ = \max \{0, x_i\} \text{ and } x_i^- = \max \{0, -x_i\} \quad \text{for } i = 1, \dots, n.$$

Then, (x^+, x^-) is feasible to (3.6.3). Assume that (x^+, x^-) is not optimal to (3.6.3), i.e., there exists (z^+, z^-) feasible to (3.6.3) such that

$$\sum_{i=1}^n f_i(z_i) \leq \sum_{i=1}^n f_i(z_i^+ + z_i^-) < \sum_{i=1}^n f_i(x_i^+ + x_i^-) = \sum_{i=1}^n f_i(x_i)$$

where $z = z^+ - z^-$ and z is feasible to (3.6.1). This is a contradiction since x is optimal to (3.6.1). \square

Now, consider the subproblems of the two problems equivalent to (3.6.1) at a point (x^+, x^-) or $x = x^+ - x^-$

$$\begin{aligned} \min \quad & \sum_{i=1}^n \delta f_i(|x_i|) y_i^+ + \sum_{i=1}^n \delta f_i(|x_i|) y_i^- \\ \text{s.t.} \quad & A (y_i^+ - y_i^-) = b \\ & y_i^+, y_i^- \geq 0 \quad i = 1, \dots, n \end{aligned} \tag{3.6.5}$$

$$\begin{aligned} \min \quad & \sum_{i=1}^n \delta f_i(x_i^+) y_i^+ + \sum_{i=1}^n \delta f_i(x_i^-) y_i^- \\ \text{s.t.} \quad & A (y_i^+ - y_i^-) = b \\ & y_i^+, y_i^- \geq 0 \quad i = 1, \dots, n. \end{aligned} \tag{3.6.6}$$

Note that (3.6.5) and (3.6.6) always have a bounded solution since the cost coefficients are always nonnegative due to Assumption 3.6.1.

In theory, we can solve Problem (3.6.1) by solving either Problem (3.6.3) or (3.6.4) instead. However, this is not quite efficient since the number of variables doubles in converting Problem (3.6.1) to (3.6.3) or (3.6.4). What would be more efficient is to replace Step 1 of RSD by the following

Step 1: (Subproblem)

Solve

$$\begin{aligned} \min \quad & \sum_{i=1}^n \delta f_i(|x_i^k|) y_i^+ + \sum_{i=1}^n \delta f_i(|x_i^k|) y_i^- \\ \text{s.t.} \quad & A (y_i^+ - y_i^-) = b \\ & y_i^+, y_i^- \geq 0 \quad i = 1, \dots, n \end{aligned}$$

or

$$\begin{aligned} \min \quad & \sum_{i=1}^n \delta f_i(x_i^{k+}) y_i^+ + \sum_{i=1}^n \delta f_i(x_i^{k-}) y_i^- \\ \text{s.t.} \quad & A (y_i^+ - y_i^-) = b \\ & y_i^+, y_i^- \geq 0 \quad i = 1, \dots, n \end{aligned}$$

where $x_i^{k+} = \max \{0, x_i^k\}$ and $x_i^{k-} = \max \{0, -x_i^k\}$.

If $\sum_{i=1}^n \delta f_i(|x_i^k|) \{(y_i^{k+} - x_i^{k+}) + (y_i^{k-} - x_i^{k-})\} \geq 0$

[or $\sum_{i=1}^n \delta f_i(x_i^{k+}) (y_i^{k+} - x_i^{k+}) + \sum_{i=1}^n \delta f_i(x_i^{k-}) (y_i^{k-} - x_i^{k-}) \geq 0$],

x^k is a solution and terminate.

Otherwise, set $y^k = y^{k+} - y^{k-}$ and continue with options (i) and (ii) of RSD.

The convergence of the modified RSD algorithm follows the standard convergence proof, given that y^k in Step 1 does in fact provide a descent direction. The theorem below demonstrates that both Subproblems (3.6.5) and (3.6.6) do provide a descent direction.

Theorem 3.6.2: For a given $x \in S$, the following hold

- i) $\sum_{i=1}^n \delta f_i(|x_i|)\{(y_i^+ - x_i^+) + (y_i^- - x_i^-)\} \geq \sum_{i=1}^n \delta f_i(x_i)(y_i - x_i).$
- ii) $\sum_{i=1}^n \delta f_i(x_i^+)(y_i^+ - x_i^+) + \sum_{i=1}^n \delta f_i(x_i^-)(y_i^- - x_i^-) \geq \sum_{i=1}^n \delta f_i(x_i)(y_i - x_i).$

Proof: Define $I_+ = \{i : x_i \geq 0, i = 1, \dots, n\}$ and

$I_- = \{i : x_i < 0, i = 1, \dots, n\}$. Then, (i) follows from

$$\begin{aligned} & \sum_{i=1}^n \delta f_i(|x_i|)\{(y_i^+ - x_i^+) + (y_i^- - x_i^-)\} - \sum_{i=1}^n \delta f_i(x_i)(y_i - x_i) \\ &= \sum_{i \in I_+} \delta f_i(|x_i|)(y_i^+ - x_i^+) + \sum_{i \in I_-} \delta f_i(|x_i|)y_i^+ \\ & \quad + \sum_{i \in I_+} \delta f_i(|x_i|)y_i^- + \sum_{i \in I_-} \delta f_i(|x_i|)(y_i^- - x_i^-) \\ & \quad - \left[\sum_{i \in I_+} \delta f_i(|x_i|)(y_i^+ - x_i^+) + \sum_{i \in I_-} \delta f_i(|x_i|)(y_i^- - x_i^-) \right. \\ & \quad \left. - \sum_{i \in I_+} \delta f_i(|x_i|)y_i^- - \sum_{i \in I_-} \delta f_i(|x_i|)y_i^+ \right] \end{aligned}$$

$$= 2 \left\{ \sum_{i \in I_+} \delta f_i(|x_i|) y_i^- \right\} + 2 \left\{ \sum_{i \in I_-} \delta f_i(|x_i|) y_i^+ \right\} \geq 0$$

and (ii) follows from

$$\begin{aligned} & \sum_{i=1}^n \delta f_i(x_i^+)(y_i^+ - x_i^+) + \sum_{i=1}^n \delta f_i(x_i^-)(y_i^- - x_i^-) - \sum_{i=1}^n \delta f_i(x_i)(y_i - x_i) \\ &= \sum_{i \in I_+} \delta f_i(|x_i|)(y_i^+ - x_i^+) + \sum_{i \in I_-} \delta f_i(|x_i|)(y_i^- - x_i^-) \\ & \quad - \left[\sum_{i \in I_+} \delta f_i(|x_i|)(y_i^+ - x_i^+) + \sum_{i \in I_-} \delta f_i(|x_i|)(y_i^- - x_i^-) \right. \\ & \quad \left. - \sum_{i \in I_+} \delta f_i(|x_i|) y_i^- - \sum_{i \in I_-} \delta f_i(|x_i|) y_i^+ \right] \\ &= \sum_{i \in I_+} \delta f_i(|x_i|) y_i^- + \sum_{i \in I_-} \delta f_i(|x_i|) y_i^+ \geq 0. \end{aligned}$$

Assume that (y^+, y^-) and (z^+, z^-) are solutions for (3.6.5) and (3.6.6), respectively. The theorem below shows that (3.6.6) gives a steeper direction than (3.6.5).

Theorem 3.6.3: For a given $x \in S$, the following hold

$$\sum_{i=1}^n \delta f_i(x_i)(z_i - x_i) \leq \sum_{i=1}^n \delta f_i(x_i)(y_i - x_i)$$

where $y = y^+ - y^-$ and $z = z^+ - z^-$.

Proof: It is sufficient to show that

$$\sum_{i=1}^n \delta f_i(x_i) z_i \leq \sum_{i=1}^n \delta f_i(x_i) y_i.$$

Without loss of generality, assume that $x_i \geq 0$ for $i = 1, \dots, n$. If not, we can define a new node-arc incidence matrix \hat{A} changing the

sign of the columns corresponding to the negative variables.

Since (y^+, y^-) is optimal to (3.6.5)

$$\sum_{i=1}^n \delta f_i(x_i)(y_i^+ + y_i^-) \leq \sum_{i=1}^n \delta f_i(x_i)(z_i^+ + z_i^-). \quad (3.6.7)$$

Since (z^+, z^-) is optimal to (3.6.6) and $x_i \geq 0$ implies that $\delta f_i(x_i^-) = 0$ for $i = 1, \dots, n$. It follows that

$$\begin{aligned} \sum_{i=1}^n \delta f_i(x_i^+)z_i^+ &\leq \sum_{i=1}^n \delta f_i(x_i^+)y_i^+ \\ \sum_{i=1}^n \delta f_i(x_i^-)z_i^- &\leq \sum_{i=1}^n \delta f_i(x_i^-)y_i^- \\ \sum_{i=1}^n \delta f_i(x_i)(z_i^+ - z_i^-) &\leq \sum_{i=1}^n \delta f_i(x_i)(y_i^+ - y_i^-). \end{aligned} \quad (3.6.8)$$

From (3.6.7) and (3.6.8)

$$\begin{aligned} \sum_{i=1}^n \delta f_i(x_i)(z_i^+ - z_i^-) &\leq \sum_{i=1}^n \delta f_i(x_i)(y_i^+ - z_i^-) + \sum_{i=1}^n \delta f_i(x_i)(z_i^+ + z_i^-) \\ &\quad - \sum_{i=1}^n \delta f_i(x_i)(y_i^+ + y_i^-) \\ &= \sum_{i=1}^n \delta f_i(x_i)z_i^+ - \sum_{i=1}^n \delta f_i(x_i)y_i^- \\ &\leq \sum_{i=1}^n \delta f_i(x_i)y_i^+ - \sum_{i=1}^n \delta f_i(x_i)y_i^- \\ &= \sum_{i=1}^n \delta f_i(x_i)(y_i^+ - y_i^-). \end{aligned}$$

3.7 Computational Results

3.7.1 Colville Problems

The three linearly constrained Colville problems, numbers 1, 4 and 7, were solved on a VAX 11/750 computer. For the formulations, see Colville [1968] or the text by Himmelblau [1972]. The RSD results corresponding to the projected Newton and quasi-Newton methods are displayed in Tables 3-1 and 3-2, respectively, showing the results of 50 iterations for different values of r . The first of these, the "Shell Problem," has five nonnegative variables and ten additional linear constraints, the objective function is the sum of linear, quadratic and cubic terms. The optimal objective value is -32.34868, as verified by solving the "Shell Dual", Colville problem 2. When $r=2$, both methods obtain the optimal value in four iterations, whereas the Frank-Wolfe method exhibits slow convergence [Wolfe 1970]: to produce the objective accurate to seven digits, the projected Newton version required over 25000 iterations! The CPU times are approximate because the VAX clock does not account for the system load.

Results for "Wood's Problem" are in Tables 3-1(b) and 3-2(b). It has four variables in the objective which is a nonconvex polynomial of degree four. The only constraints are simple upper and lower bounds on all variables. The unconstrained minimum occurs within this region and the optimal value is zero. Furthermore, there are both easy and hard starting points for the problem. The start shown is "hard" in the sense that some methods will converge to a nonoptimal point (see Colville

[1968]). Since the projected Newton method requires a positive definite Hessian matrix in solving the master, the diagonal elements of the true Hessian were (heuristically) increased at any master iteration where this was not the case. It is somewhat remarkable that all four versions converged to the optimal solution. However, for $r=1$ and $r=2$, the total number of iterations to reduce the objective function below 10^{-7} was 16880 and 6920, respectively. For $r=3$, it required 20 iterations, and for $r=4$, just 8. The projected quasi-Newton method performed much better than the projected Newton. For $r=2$, it required 4 iterations, and for $r=3$, only 3.

The final Colville problem has 16 variables, with eight linear constraints plus lower and upper bounds on all variables. The objective (to be maximized) is fourth order. Tables 3-1(c) and 3-2(c) show that both methods are comparable and demonstrate that, again, the performance of RSD improves steadily as r increases.

3.7.2 Traffic Assignment Problems

Many models very much like the model presented here have been developed over the last 35 years. The earliest examples of this kind are due to Beckmann [1951] and Wardrop [1952], and later by Dafermos and Sparrow [1969], Nguyen [1974], LeBlanc et al. [1975] and many others. The model considered is the following multicommodity network flow problem

Table 3-1: RSD Results for Three Colville Problems (Newton method).

Objective Function Value					
	Iter.	r=1(FW)	r=2		
(a) <u>Shell Problem</u>	1	-15.55724	-15.55724		
	2	-21.66206	-27.72511		
Init. Solution:	3	-24.51198	-32.28357		
(0,0,0,0,1)	4	-27.49879	-32.34868		
	5	-28.59594			
	10	-31.05066			
	20	-31.93449			
	30	-32.10358			
	50	-32.21180			
Total CPU sec.		4.29	0.82		
Total # of Proj.		92	14		
Iter. to 7 digits accuracy		>25000	4		
	Iter.	r=1(FW)	r=2	r=3	r=4
(b) <u>Wood's Problem</u>	1	8.839690	8.839690	8.839690	8.839690
	2	8.465989	7.877027	7.877027	7.877027
Init. Solution:	3	8.254216	7.876967	7.876967	7.876967
(-3,-1,-3,-1)	4	8.119669	7.876967	7.876967	7.876966
	5	8.036823	7.876967	7.873999	6.261434
	8	7.924362	7.876967	6.959996	0.000000
	10	7.898680	7.876967	1.303893	
	20	7.877552	7.876967	0.000000	
	30	7.876988	7.876967		
	50	7.876967	7.876967		
Total CPU sec.		3.53	18.23	6.30	2.81
Total # of Proj.		130	239	149	67
Iter. to 7 digits accuracy		16880	6920	20	8
	Iter.	r=1(FW)	r=2	r=3	r=4
(c) <u>Gauthier's Prob.</u>	1	-245.6341	-245.6341	-245.6341	-245.6341
	2	-245.2229	-245.2223	-245.2223	-245.2223
Init. Solution:	3	-245.0936	-245.0846	-245.0147	-245.0147
(0,0,...,0)	4	-245.0524	-245.0252	-245.0043	-244.9894
	5	-245.0354	-244.9899	-244.9917	
	6	-245.0325	-244.9896	-244.9894	
	10	-245.0020	-244.9895		
	11	-244.9984	-244.9894		
	20	-244.9909			
	30	-244.9897			
	50	-244.9896			
Total CPU sec.		19.21	6.91	5.68	4.16
Total # of Proj.		53	18	13	8
Iter. to 7 digits accuracy		158	11	6	4

Table 3-2: RSD Results for Three Colville Problems (Quasi-Newton method).

Objective Function Value					
	Iter.	r=1(FW)	r=2		
(a) <u>Shell Problem</u>	1	-15.55724	-15.55724		
	2	-21.66206	-27.72511		
Init. Solution:	3	-24.51198	-32.28357		
(0,0,0,0,1)	4	-27.49879	-32.34868		
	5	-28.59594			
	10	-31.05066			
	20	-31.93449			
	30	-32.10356			
	50	-32.21175			
Total CPU sec.		4.48	0.80		
Total # of Proj.		165	19		
Iter. to 7 digits accuracy		-	4		
	Iter.	r=1(FW)	r=2	r=3	
(b) <u>Wood's Problem</u>	1	158.188639	158.188639	158.188639	
	2	144.103277	0.961506	0.961506	
Init. Solution:	3	21.531098	0.226265	0.000000	
(-3,-1,-3,-1)	4	11.135727	0.000000		
	5	9.862158			
	10	8.061648			
	20	7.880695			
	30	7.877078			
	50	7.876967			
Total CPU sec.		4.53	0.53	0.49	
Total # of Proj.		201	20	15	
Iter. to 7 digits accuracy		-	4	3	
	Iter.	r=1(FW)	r=2	r=3	r=4
(c) <u>Gauthier's Prob.</u>	1	-245.6341	-245.6341	-245.6341	-245.6341
	2	-245.2229	-245.2223	-245.2223	-245.2223
Init. Solution:	3	-245.0936	-245.0846	-245.0147	-245.0147
(0,0,...,0)	4	-245.0524	-245.0252	-245.0043	-244.9896
	5	-245.0354	-244.9899	-244.9917	-244.9894
	6	-245.0325	-244.9896	-244.9894	
	8	-245.0082	-244.9894		
	10	-245.0020			
	20	-244.9909			
	30	-244.9897			
	50	-244.9895			
Total CPU sec.		17.75	5.39	4.77	4.71
Total # of Proj.		95	32	26	23
Iter. to 7 digits accuracy		-	8	6	5

$$\begin{aligned}
 \min \quad & \sum_a \int_0^{x_a} f_a(t) dt \\
 \text{s.t.} \quad & x_a = \sum_k x_a^k \quad \text{for all } a \\
 & A x^k = b^k \quad \text{for all } k \in D \\
 & x^k \geq 0 \quad \text{for all } k \in D
 \end{aligned}$$

where D is the set of origin/destinations (commodities), x_a^k is the flow of the k^{th} commodity on link a , x_a is the aggregate flow on link a , x^k is the vector of flows of the k^{th} commodity, and $f_a(\bullet)$ is a convex nondecreasing function that represents the travel time on link a . The optimal solution of this problem is a set of flows satisfying Wardrop's "user-equilibrium principle". According to this principle, the cost (time) of every utilized path between an origin and a destination is a constant no greater than the cost of any nonutilized path. Of course, the cost of a path is the sum of the link costs for links in the path. Table 3-3 summarizes the results on four traffic assignment problems from the literature, two of which are based on street networks for the Canadian cities of Hull and Winnipeg. In the Table the size of each network is displayed as nodes/links/centroids. A centroid is a node which is either an origin or a destination of some fixed travel demand. The absence of upper bounds on the variables causes the linear subproblem to decompose into a set of shortest path problems, one for each commodity.

The quantity LB associated with each problem is the largest lower bound generated by RSD using the tangent plane inequality (equivalently,

the duality gap, see Hearn [1982]). All problems were run in single precision FORTRAN on an IBM 3081D which has an accurate CPU clock. The reason for termination of each run is shown in the final column, whether maximum allowed iterations (IT), maximum CPU seconds (CPU), relative error within the present tolerance (RE), or three consecutive master iterations could not decrease $f(\bullet)$ with a step of at least 10^{-4} in the projected direction (ND).

Table 3-4 compares the performance of RSD with the projected Newton method (RSD_1) and the RSD code of Lawphongpanich and Hearn [1983] (RSD_2) for the two small traffic networks of Table 3-3. It is clear that RSD_1 outperforms RSD_2 in terms of total number of projections, CPU time, and accuracy of the objective function value.

In the traffic assignment problems, the majority of the effort (80% - 90%) is spent on the shortest path calculations. Using the number of shortest path calculations as the sole criterion, RSD compares very favorably with other algorithms (see Table 3-5). In fact, there is only one instance in which another method, RESTRICTION on the Hull network, outperforms RSD. However, as described in Guelat [1983], RESTRICTION can be considered as a variant of RSD which allows for restarting at every k^{th} iteration.

Considering the size of the Hull and Winnipeg networks, it may be surprising that RSD performs well with a relatively small value of r . This is partially explained by the fact that solutions to traffic assignment problems involve a small number of utilized paths. Since there is a one-to-one correspondence between paths and extreme points, a

Table 3-3: RSD Results for Traffic Assignment Problems (Newton Method).

Problem	r	CPU Sec.	No. Iter.	No. Proj.	Final Obj.	% Relative Error	Term.
NINE-NODE (9/18/4) (LB = 1453.13) Lawphongpanich and Hearn [1983]	1	1.13	50	55	1457.54	0.303	IT = 50
	2	1.19	50	74	1454.50	0.0943	IT = 50
	3	1.22	39	66	1453.13	0.000	RE \leq 1.E-6
	4	1.18	38	74	1453.25	0.00826	ND
	5	0.47	14	28	1453.14	0.000688	ND
DUPUIS (13/19/4) (LB = 85027.6) Lawphongpanich and Hearn [1983]	1	1.19	50	50	85167.6	0.165	IT = 50
	2	1.21	50	85	85072.5	0.0528	IT = 50
	3	0.87	32	60	85038.8	0.0132	ND
	4	0.22	7	13	86027.6	0.000	RE \leq 1.E-6
HULL (501/798/23) (LU = 34805.74) Florian et al. [1983]	1	33.23	100	109	34817.27	0.0331	IT = 100
	3	37.17	89	150	34808.72	0.00856	RE \leq 1.E-4
	5	20.50	41	95	34808.88	0.00902	RE \leq 1.E-4
	7	15.97	31	74	34808.42	0.00769	RE \leq 1.E-4
	9	19.05	35	76	34808.25	0.00721	RE \leq 1.E-4
	11	17.36	32	66	34808.07	0.00669	RE \leq 1.E-4
	≥ 12	15.08	27	62	34808.18	0.00701	RE \leq 1.E-4
WINNIPEG (1052/2836/147) (LU = 890958.) Florian et al. [1983]	1	485.37	73	80	893068.	0.237	CPU = 500.
	4	490.33	69	136	891968.	0.113	CPU = 500.
	9	487.72	58	135	891726.	0.0862	CPU = 500.
	14	485.37	60	105	891706.	0.0840	CPU = 500.

Table 3-5: Number of Shortest Path Calculations to achieve the Relative Error.

Algorithm		% Relative Error					
		DUPUIS		HULL		WINNIPEG	
		.10%	.05%	.10%	.05%	1.0%	0.5%
RSD	r=1	87	NA	43	71	29	43
	r=2	22	55				
	r=3	11	11	15	19		
	r=4	6	6			18	26
	r=5			12	16		
	r=9			12	16	15	20
Guelat [1983]	FW	NA	NA	37	70	29	45
	Partan	11	28	15	21	18	27
	Restriction	9	9	10	13	17	25
	Quadratic Approx.			22	27	25	36
Dembo and Tulowitzki [1983]*	FW			65	171	46	80
	TQP-FW			32	56	35	52
	Partan			23	41	34	47
	TQP-PT			32	60	31	48

NA = The algorithm terminated before the indicated Relative Error was obtained.

Restriction = A strategy similar to RSD with restart every 6 iterations.
Quadratic Approx. = Guelat's variation of Truncated Quadratic Prog.

TQP-FW = Truncated Quadratic Programming using FW on the Quadratic Subproblem.

TQP-PT = Truncated Quadratic Programming using Partan on the Quadratic Subproblem.

* = The Relative Error as reported in Dembo and Tulowitzki [1983] is based on the lower bound calculated at the current iterate.

small number of utilized paths implies a small number of extreme points in the optimal face.

3.7.3 Electrical Network Problems

These models arise in the study of conducting fibers (sticks) imbedded in an insulating polymer [Balberg et al. 1983]. The conductivity in the composite is induced when the sticks form a continuous network from one boundary to its opposite. In these networks, nodes (junctions) represent the sticks and the boundaries, and arcs (resistors) represent the intersections formed by overlapping sticks. It is well known that the equilibrium conditions of an electrical network are attained as the total energy loss is minimized (see, e.g., Cooper and Kennington [1977]). These networks of resistors have a single origin and a single destination. If the total flow is assumed to be one unit and the resistance in each arc is also one unit, then the total energy loss equals the effective resistance of the network. The formulation of the problem is

$$\begin{aligned} \min \quad & \sum_{i=1}^n x_i^2 \\ \text{s.t.} \quad & Ax = b \end{aligned}$$

where x_i denotes the current on arc i .

Table 3-6 summarizes the results for RSD on five large-scale electrical networks taken from the studies of Balberg et al. [1983]. These problems are of interest in this study because they are pathological (to RSD) in the sense that many paths from the origin to

the destination will be utilized, and this implies many extreme points defining the optimal face. In fact, it is easy to verify that the number grows exponentially with the size of the network. The lower bounds were obtained by solving the unconstrained quadratic dual problem by the conjugate gradient method (see Subsection 4.2.2). While the relative error is shown to decrease with increasing r , the increase in CPU time (IBM 3081D) does not justify the decrease and the basic Frank-Wolfe method is not significantly improved upon. For very large values of $r=30$ and $r=50$, this performance further demonstrates that the computational burden of the larger master problem is too great relative to the effort required to solve the subproblem.

Subsection 4.2.2 shows that the dual of this problem can be solved much more efficiently by the conjugate gradient method in conjunction with sparse matrix techniques.

3.7.4 Water Distribution Problems

These models deal with determining the steady state flows and pressures in a water distribution network. Collins et al. [1978] give a precise discussion and formulation of these models as optimization problems. The formulation is

$$\begin{aligned} \min \quad & \sum_{i=1}^q f_i(x_i) + \sum_{j=1}^p c_j y_j \\ \text{s.t.} \quad & A_1 x + A_2 y = b \\ & 1 \leq x \leq u \end{aligned}$$

Table 3-6: RSD Results for Electrical Network Problems (Newton Method).

Problem	r	CPU Sec.	No. Iter.	No. Proj.	Final Obj.	% Relative Error	Term.
S-1 (237/454/2) (LB = 6.934391) Balberg et al.[1983]	1	4.44	100	100	6.940507	0.0882	IT = 100
	2	4.70	100	100	6.939835	0.0785	IT = 100
	4	5.43	100	100	6.937602	0.0463	IT = 100
S-2 (722/1412/2) (LB = 3.124562) Balberg et al.[1983]	1	9.26	100	100	3.147969	0.749	IT = 100
	2	10.11	100	100	3.146704	0.709	IT = 100
	4	12.58	100	100	3.141849	0.553	IT = 100
S-3 (952/1686/2) (LB = 11.179775) Balberg et al.[1983]	1	10.29	100	100	11.213708	0.304	IT = 100
	2	11.70	100	100	11.208403	0.256	IT = 100
	4	14.12	100	100	11.200962	0.190	IT = 100
S-4 (852/2264/2) (LB = 1.566194) Balberg et al.[1983]	1	12.69	100	100	1.598765	2.08	IT = 100
	2	15.10	100	100	1.598158	2.04	IT = 100
	4	19.01	100	100	1.597045	1.97	IT = 100
S-5 (902/3699/2) (LB = 0.467484) Balberg et al.[1983]	1	16.73	100	100	0.541306	15.8	IT = 100
	2	20.72	100	100	0.540259	15.6	IT = 100
	4	26.28	100	100	0.539956	15.5	IT = 100

where some of the $f_i(\bullet)$ are of the form

$$\begin{aligned} f_i(x_i) &= \int_0^{x_i} -(b_i(a_i - t^2))^{\frac{1}{2}} dt \\ &= -\frac{b_i^{\frac{1}{2}}}{2} \left[x_i(a_i - x_i^2)^{\frac{1}{2}} + a_i \sin^{-1} \frac{x_i}{a_i^{\frac{1}{2}}} \right] \end{aligned}$$

and the rest of the $f_i(\bullet)$ are of the form

$$f_i(x_i) = k_i |x_i|^{2.85}$$

where a_i, b_i and k_i are constants.

We have tested on the IBM 3081D the Dallas water distribution model (see Table 3-7) which has been employed as a test problem for a number of studies [Collins et al. 1978, Dembo and Klincewicz 1981, Beck et al. 1983, Dembo 1983, Klincewicz 1983, and Kamesam and Meyer 1984]. This problem has the potential for having a large number of extreme points in the optimal face. Thus, one can expect a result similar to that of the electrical networks. Nevertheless, it is encouraging that the objective function value for $r=4$ is lower than the previous best value -206175.2 [Kamesam and Meyer 1984] and was obtained in approximately one minute of CPU time. It should be noted that the inaccuracies in the approximating functions of the Dallas problem induce negative cycles in the solution and preclude the calculation of a lower bound in the standard manner [Kennington 1984].

Table 3-7: RSD Results for the Water Distribution Problem (Newton Method).

Problem	r	CPU Sec.	No. Iter.	No. Proj.	Final Obj.	Term.
DALLAS	1	56.14	88	131	-205676.7	ND
(667/1796/425)	2	54.88	90	90	-205888.4	ND
Collins et al.	3	53.08	87	87	-205853.3	ND
[1978]	4	61.68	103	103	-206179.8	ND
	5	46.62	73	73	-205780.8	ND

CHAPTER FOUR

DUAL METHODS FOR SEPARABLE STRICTLY CONVEX QUADRATIC NETWORK PROBLEMS

4.1 Introduction

The electrical network problems discussed in Subsection 3.6.3 are a clear example of a case where the RSD algorithm does not perform well in comparison to other methods such as the FW algorithm (special case of RSD when $r=1$). This is due to the fact that the dimension of the optimal face is close to n (the number of variables) and, therefore, in order to be able to generate a simplex containing the optimal solution, the parameter r has to be very large. This then makes the repeated solving of the master problem ineffective in terms of CPU time compared to the CPU time required to solve the linear subproblem (a simple shortest path calculation). However, the dual of this problem is an unconstrained convex quadratic program, the Hessian is sparse, and the structure is simple due to the network constraints.

This chapter is concerned with the application of conjugate gradient based methods to solve dual formulations of separable strictly convex quadratic network problems. Two different cases are considered: the first one specializes the classical conjugate gradient method of Hestenes and Stiefel [1952] and a preconditioned version to networks with undirected arcs; and the second applies a dual based ascent method

to networks where the flows on the arcs are subject to upper and lower bounds. The ascent directions of this method are also generated by conjugate gradient formulas.

It is important to point out that, although this chapter is concerned with pure network problems, the extension of these methods to generalized networks is straightforward and can be done in a similar manner.

4.2 CASE I: Networks with Undirected Arcs

Consider the following problem

$$\begin{aligned} \min \quad & f(x) = \frac{1}{2} x^T D x + c^T x \\ \text{s.t.} \quad & A x = b \end{aligned} \quad (4.2.1)$$

where $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, D is a diagonal positive definite $n \times n$ matrix, and A is an $m \times n$ node-arc incidence matrix of a graph $G(V, E)$.

The Wolfe dual [Wolfe 1961] of (4.2.1) is

$$\begin{aligned} \max \quad & \frac{1}{2} x^T D x + c^T x + \mu^T (A x - b) \\ \text{s.t.} \quad & D x + c + A^T \mu = 0 \end{aligned} \quad (4.2.2)$$

where $\mu \in \mathbb{R}^m$ is the vector of dual variables.

We can eliminate the x variables by using the relation

$$x = -D^{-1}(A^T \mu + c)$$

and the dual problem becomes

$$\min g(\mu) = \frac{1}{2} \mu^T A D^{-1} A^T \mu + (D^{-1} A^T c + b)^T \mu + \frac{1}{2} c^T D^{-1} c \quad (4.2.3)$$

where the Hessian matrix is $H = AD^{-1}A^T$ and the linear coefficients are $\bar{b} = D^{-1}A^Tc + b$. H and \bar{b} can be computed directly by using the network data structure as follows

$$H_{ii} = \sum_{j=1}^n (1/D_{ij}) A_{ij}^2 = \sum_{j \in E_i} 1/D_{ij} \quad i = 1, \dots, m$$

where $E_i = \{j : (i,j) \in E\}$.

$$H_{ij} = \sum_{k=1}^n (1/D_{ik}) A_{ik} A_{jk} = \begin{cases} -1/D_{ij} & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \quad i \neq j$$

$$\bar{b}_{ij} = (1/D_{ij})(c_i - c_j) + b_{ij} \quad (i,j) \in E.$$

Note that the matrix H is positive semidefinite since $\mu^T H \mu = (D^{-\frac{1}{2}} A^T \mu)^T (D^{-\frac{1}{2}} A^T \mu) = \|D^{-\frac{1}{2}} A^T \mu\|^2 \geq 0$ for all $\mu \in R^m$. This implies that the dual problem (4.2.3) is a convex quadratic program. Therefore, μ^* is a solution of (4.2.3) if and only if $\delta g(\mu^*) = 0$. Thus

$$\delta g(\mu^*) = H \mu^* + \bar{b} = 0. \quad (4.2.4)$$

The solution of this system of linear equations is discussed in the next subsection.

4.2.1 Conjugate Gradient Method

There is a substantial class of large-scale systems of linear equations for which the coefficient matrix, H , is sparse and its elements may be generated by some simple formula. For instance, in (4.2.4) the elements of H are easily determined by the structure of the

network. Therefore, it is desirable to solve such linear systems by iterative methods that never alter H . Such a method is the conjugate gradient (CG) algorithm originally introduced by Hestenes and Stiefel [1952] to solve square systems of linear equations, and applied to nonlinear problems by Fletcher and Reeves [1964], Polak and Ribiere [1969], and others. Below, we present this algorithm and show how the matrix operations can be carried out for the linear system (4.2.4).

CG Algorithm

Step 0: Set $u^0 = 0$, $r^0 = \bar{b}$, $d^0 = -\bar{b}$ and $k = 0$.

Step 1: Compute

$$\alpha^k = \frac{-r^{kT} d^k}{d^{kT} H d^k}$$

$$u^{k+1} = u^k + \alpha^k d^k$$

$$r^{k+1} = H u^{k+1} - \bar{b}$$

and go to 2.

Step 2: Compute

$$\beta^k = \frac{r^{k+1T} r^{k+1}}{r^{kT} r^k}$$

$$d^{k+1} = -r^{k+1} + \beta^k d^k$$

and go to 3.

Step 3: If $\|r^{k+1}\| \leq \epsilon$, terminate.

Otherwise, set $k = k + 1$ and go to 1.

In theory the directions d^k are H-conjugate, i.e., $d^k H d^{k'} = 0$ for all $k \neq k'$, and μ^k minimizes the function in (4.2.3) on the manifold defined by $\mu^0, d^0, \dots, d^{k-1}$.

Note that the CG algorithm is efficient in the sense that only two expressions with matrix operations are involved: $d^{kT} H d^k$ and $H \mu^k$. We show below that these expressions can be carried out very easily by using only node-arc network information.

i) $d^{kT} H d^k$:

$$\begin{aligned} d^{kT} H d^k &= d^{kT} A D^{-1} A^T d^k \\ &= \sum_{(i,j) \in E} (1/D_{ij}) (d_i^k - d_j^k)^2 \end{aligned}$$

ii) $H \mu^k$:

$$\begin{aligned} (H \mu^k)_i &= \sum_{j=1}^n H_{ij} \mu_j^k \\ &= H_{ii} \mu_i^k - \sum_{j \in E_i} D_{ij} \mu_j^k \quad i = 1, \dots, m \end{aligned}$$

where $E_i = \{j : (i,j) \in E\}$.

Substituting (i) and (ii) in Step 1 of the CG algorithm, one obtains a version of the algorithm to solve (4.2.4) which does not involve an explicit representation of H. In terms of storage requirements, the $m \times m$ matrix H has been replaced by a vector of size m to store the diagonal elements of H and a $3 \times n$ matrix to store the information of the arc set E. This exploitation of the sparsity of H is similar to the approach suggested by Dembo [1979] and by Klincewicz [1983] in the adaptation of Newton methods to network problems.

In general, if the matrix H has $q \leq m$ distinct eigenvalues, the CG algorithm will converge in q steps with exact arithmetic. With inexact arithmetic, however, the accuracy of the solution and the rate of convergence is determined from the eigenvalue structure of H (see, e.g., Axelsson [1977] or Jennings [1977]). Jennings shows that, if H is positive definite and there is no rounding error, the number of iterations for the CG method to achieve a certain accuracy depends on the condition number of H , i.e., $\text{cond } H = e_n/e_1$, where e_n and e_1 are the largest and smallest eigenvalues of H . When most of the eigenvalues are in a relatively small interval compared to the whole spectrum, $e_n - e_1$, the method will reach the same accuracy more rapidly. Jennings shows that if the stopping criteria of the CG method is $\|r^k\|/\|b\| \leq \epsilon$, the number of iterations to achieve a tolerance ϵ is

$$k \leq \frac{\cosh^{-1}(1/\epsilon)}{\cosh^{-1}[(e_n + e_1)/(e_n - e_1)]}$$

and if $\epsilon \ll 1$ and $e_n \gg e_1$, the above inequality can be approximated by

$$k \leq \frac{1}{2} (e_n/e_1)^{\frac{1}{2}} \ln(2/\epsilon).$$

These arguments show that it will be helpful if the system of equations (4.2.4) could be transformed into an equivalent system with better conditioning. First of all, the matrix H is required to be positive definite. This can be easily done by eliminating the first row and column of H and setting the associated variable to zero, i.e., $u_1 = 0$. Secondly, to improve the conditioning there are two methods:

scaling and preconditioning. An example of scaling is to divide each entry H_{ij} of H by the norm of the i^{th} row and the norm of the j^{th} column, and scaling the vectors μ and \bar{b} accordingly. Although this method is heuristic, if H has entries of different magnitude, the scaling can often improve the condition number of H [Nickel and Tolle 1984]. In applying preconditioning, it is necessary to find a matrix C , known as the preconditioning matrix, such that the condition number of $C^{-1}H$ is better than that of H . The ideal is to choose $C = H$, but the computation of H^{-1} is more difficult than solving (4.2.4).

The preconditioned conjugate gradient (PCG) method [Axelsson 1977] presented below does not form any inverse in computation but, in theory, it solves a system of linear equations equivalent to (4.2.4) given a positive definite preconditioning matrix C .

PCG Algorithm

Step 0: Set $\mu^0 = 0$, $r^0 = 0$, and $k = 0$.

Step 1: Solve

$$C \hat{r}^0 = r^0$$

$$\text{and set } d^0 = -\hat{r}^0.$$

Step 2: Compute

$$\alpha^k = \frac{r^{kT} \hat{r}^k}{d^{kT} H d^k}$$

$$\mu^{k+1} = \mu^k + \alpha^k d^k$$

$$r^{k+1} = H \mu^{k+1} - \bar{b}$$

Step 3: Solve

$$\hat{C}r^{k+1} = r^{k+1}$$

and compute

$$\beta^k = \frac{r^{k+1T}\hat{r}^{k+1}}{r^{kT}\hat{r}^k}$$

$$d^{k+1} = -\hat{r}^{k+1} + \beta^k d^k$$

Step 4: If $(r^{k+1}, \hat{r}^{k+1})^{\frac{1}{2}} \leq \epsilon$, terminate.

Otherwise, set $k = k + 1$ and go to 2.

Hestenes [1980] shows that the PCG algorithm is equivalent to applying the basic CG to a system of linear equations whose coefficient matrix is $C^{-\frac{1}{2}}HC^{-\frac{1}{2}}$, which is similar to $C^{-1}H$.

The PCG algorithm has an additional step with respect to the basic algorithm, which requires the solution of the system $\hat{C}r = r$. Thus, this method will be practical only if C is close to H in the sense that the condition number of $C^{-1}H$ is smaller than that of H , and $\hat{C}r = r$ is easy to solve, i.e., requires a small number of operations and small memory requirements. Many authors (i.e., Axelsson [1977], Kershaw [1978], Munksgard [1980] and others) have studied several preconditioning matrices for different type of problems. Their results show that when the problem has special structure the PCG method can be very efficient. The matrix used in our computational experiments of Subsection 4.2.2 is a scaling matrix defined as

$$C_{ij} = \begin{cases} H_{ii} & \text{if } i = j \\ & i, j = 1, \dots, m \\ 0 & \text{otherwise.} \end{cases}$$

4.2.2 Computational Experiments

A Fortran program for the CG algorithm described in Subsection 4.2.1 was written and executed on an IBM 3081D to test the five sticks problems described in Subsection 3.6.3. The program required only 110 lines of code. By dividing the number of arcs by the number of nodes, the average ratio of the five problems is 2.48, indicating that, while these networks contain a large number of arcs, they are rather sparse.

Table 4-1 reports the number of iterations (Iter.) required to achieve a residual error ($\|r\|$) of at least 10^{-1} , 10^{-2} , 10^{-3} , 10^{-4} and 10^{-10} for the five problems. It also displays the percent relative error (% RE) calculated with respect to the solution which achieves a residual error less than 10^{-10} , the objective function value (Obj. Fun.) and the CPU time (CPU Sec.). It is important to note that the CPU time per iteration varies from 0.0037 sec. for S-1 to 0.023 sec. for S-5. This shows that the sparse matrix techniques, when incorporated into the CG algorithm, produce a very efficient method for this type of problem.

The above program was modified to implement the PCG algorithm. Table 4-2 shows the results obtained for the five problems in a similar manner as Table 4-1, except that $\|r\|$ has been replaced by $(r, \hat{r})^{\frac{1}{2}}$. In this case the CPU time per iteration varies from 0.0044 sec. for S-1 to 0.026 sec. for S-5. Although the CPU time per iteration increases with respect to the standard method, the total number of iterations has been reduced.

Table 4-3 reports a comparison between the FW algorithm and the two conjugate gradient methods (CG and PCG). Based on the CPU times, it is clear that the CG method is far superior to the FW technique. In the worst case, S-3 with RE < 10%, CG obtains a 28.4% reduction in CPU time compared to FW. On average, the reduction is approximately 72.2%. PCG requires less CPU time than CG in twenty cases, the same in one case, and more in four cases. However, the average reduction in CPU time using the PCG method is only 8.1% compared to the standard version.

4.3 Case II: Networks with Upper and Lower Bounds on the Arcs

Consider the following problem

$$\begin{aligned} \min \quad & f(x) = \frac{1}{2} x^T D x + c^T x \\ \text{s.t.} \quad & A x = b \\ & l \leq x \leq u \end{aligned} \tag{4.3.1}$$

where $c \in R^n$, $l \in (R \cup \{\pm\infty\})^n$, $u \in (R \cup \{\pm\infty\})^n$, $x \in R^n$, $b \in R^m$, D is a diagonal positive definite $n \times n$ matrix, and A is an $m \times n$ node-arc incidence matrix of a graph $G(V, E)$.

A first approach for (4.3.1) is to solve its Wolfe dual in a similar manner to the unbounded case. The Wolfe dual of (4.3.1) after the elimination of the primal variables becomes

Table 4-1: Computational Experience with the Conjugate Gradient Algorithm.

Problem	Iter.	$\ r\ $	% RE	Obj. Fun.	CPU Sec.
S-1 (237/454)	55	1.E-1	0.15	6.92401934	0.20
	76	1.E-2	0.20E-2	6.93426068	0.28
	98	1.E-3	0.30E-4	6.93438971	0.37
	110	1.E-4	0.00	6.93439180	0.41
	148	1.E-10	0.00	6.93439180	0.55
S-2 (722/1412)	59	1.E-1	0.70	3.10269593	0.63
	114	1.E-2	0.61E-2	3.12437243	1.24
	148	1.E-3	0.58E-3	3.12454425	1.61
	183	1.E-4	0.31E-4	3.12456167	2.00
	297	1.E-10	0.00	3.12456263	3.26
S-3 (952/1686)	144	1.E-1	0.30	11.14635139	1.91
	196	1.E-2	0.49E-2	11.17922315	2.60
	230	1.E-3	0.54E-4	11.17976977	3.06
	287	1.E-4	0.89E-9	11.17977526	3.82
	447	1.E-10	0.00	11.17977527	5.97
S-4 (852/2264)	53	1.E-1	1.54	1.54266758	0.83
	95	1.E-2	0.96E-2	1.56604357	1.50
	121	1.E-3	0.96E-4	1.56619310	1.92
	153	1.E-4	0.13E-4	1.56619443	2.44
	278	1.E-10	0.00	1.56619463	4.44
S-5 (902/3699)	32	1.E-1	1.86	0.45877002	0.72
	51	1.E-2	0.18E-1	0.46740087	1.16
	75	1.E-3	0.34E-3	0.46748259	1.72
	102	1.E-4	0.21E-7	0.46748414	2.35
	188	1.E-10	0.00	0.46748415	4.36

Table 4-2: Computational Experience with the Preconditioned Conjugate Gradient Algorithm.

Problem	Iter.	$(r, \tilde{r})^{\frac{1}{2}}$	% RE	Obj. Fun.	CPU Sec.
S-1 (237/454)	42	1.E-1	1.53	6.82844300	0.18
	67	1.E-2	0.37E-2	6.93413347	0.30
	75	1.E-3	0.42E-4	6.93438886	0.33
	81	1.E-4	0.17E-4	6.93439063	0.36
	105	1.E-10	0.00	6.93439180	0.46
S-2 (722/1412)	29	1.E-1	8.92	2.84571546	0.36
	74	1.E-2	0.10	3.12133159	0.94
	106	1.E-3	0.28E-3	3.12455375	1.35
	128	1.E-4	0.64E-5	3.12456248	1.64
	222	1.E-10	0.00	3.12456263	2.86
S-3 (952/1686)	80	1.E-1	2.73	10.87415707	1.26
	139	1.E-2	0.11E-1	11.17851078	2.20
	162	1.E-3	0.28E-3	11.17974343	2.57
	198	1.E-4	0.00	11.17977527	3.15
	312	1.E-10	0.00	11.17977527	4.98
S-4 (852/2264)	26	1.E-1	6.63	1.46238458	0.45
	56	1.E-2	0.11	1.56454223	1.00
	81	1.E-3	0.56E-3	1.56618582	1.46
	107	1.E-4	0.45E-5	1.56619456	1.94
	198	1.E-10	0.00	1.56619463	3.61
S-5 (902/3699)	4	1.E-1	70.90	0.13602842	0.09
	33	1.E-2	0.17	0.46667111	0.83
	54	1.E-3	0.15E-2	0.46747715	1.38
	71	1.E-4	0.86E-5	0.46748411	1.82
	136	1.E-10	0.00	0.46748415	3.51

Table 4-3: Comparison of Frank-Wolfe (FW), Conjugate Gradient (CG), and Preconditioned Conjugate Gradient (PCG) Techniques.

CPU Time in Seconds (Number of Iterations)

Problem	Method	RE < 10%	RE < 5%	RE < 3%	RE < 1%	RE < 0.5%
S-1	FW	0.36 (6)	0.49 (9)	0.58 (11)	0.88 (18)	1.31 (28)
	CG	0.15 (41)	0.16 (43)	0.16 (44)	0.18 (49)	0.19 (51)
	PCG	0.14 (33)	0.16 (36)	0.17 (39)	0.20 (46)	0.22 (50)
S-2	FW	2.62 (23)	3.48 (33)	4.34 (43)	7.71 (82)	*
	CG	0.37 (35)	0.46 (44)	0.53 (50)	0.61 (57)	0.67 (63)
	PCG	0.34 (28)	0.45 (36)	0.50 (40)	0.60 (48)	0.68 (54)
S-3	FW	1.55 (13)	2.14 (19)	2.75 (25)	4.96 (47)	7.57 (73)
	CG	1.11 (84)	1.29 (98)	1.48 (112)	1.72 (130)	1.85 (140)
	PCG	1.04 (66)	1.12 (71)	1.25 (79)	1.57 (99)	1.74 (110)
S-4	FW	6.11 (47)	8.47 (66)	*	*	*
	CG	0.53 (34)	0.62 (40)	0.72 (46)	0.89 (57)	0.98 (62)
	PCG	0.42 (24)	0.51 (29)	0.54 (31)	0.71 (40)	0.80 (45)
S-5	FW	*	*	*	*	*
	CG	0.49 (22)	0.56 (25)	0.65 (29)	0.70 (31)	0.84 (37)
	PCG	0.41 (17)	0.49 (20)	0.55 (22)	0.67 (27)	0.73 (29)

* = When FW terminates at the 100 th iteration, the indicated relative error was not achieved.

$$\begin{aligned}
 \min \quad g(\mu, v, w) = & \frac{1}{2} [\mu^T, v^T, w^T] \begin{bmatrix} AD^{-1}A^T & AD^{-1} & -AD^{-1} \\ D^{-1}A^T & D^{-1} & -D^{-1} \\ -D^{-1}A^T & -D^{-1} & D^{-1} \end{bmatrix} \begin{bmatrix} \mu \\ v \\ w \end{bmatrix} \\
 & + [c^T D^{-1}A^T + b^T, c^T D^{-1} + u^T, -c^T D^{-1} - l^T] \begin{bmatrix} \mu \\ v \\ w \end{bmatrix} + \frac{1}{2} c^T D^{-1} c \\
 \text{s.t.} \quad & v \geq 0 \\
 & w \geq 0
 \end{aligned} \tag{4.3.2}$$

where $\mu \in R^m$, $v \in R^n$ and $w \in R^n$ are the vectors of dual variables corresponding to the equality constraints, upper and lower bounds, respectively.

In recent years, several authors such as O'Leary [1980], Dembo and Tulowitzki [1983a] and Yang and Tolle [1985] have developed different versions of the CG method for convex quadratic problems subject to box constraints (upper and lower bounds only). They essentially use this iterative procedure to carry out the fundamental step of equation solving in an active-set method. Also the projected Newton method of Bertsekas (see Subsection 3.5.1) can be extended to box constrained problems. All of these iterative schemes have the property of finite convergence under exact arithmetic. The Dual Problem (4.3.2) is a special case of a box constrained problem with some lower bounds only and, therefore, these methods can be specialized to this problem taking advantage of the sparsity of the Hessian. However, since the size of

the dual problem is $m+2n$ variables, this approach may not be very promising for large networks.

For this reason, we take the approach of solving the following Lagrangian dual formulation of (4.3.1)

$$\max g(\mu) \quad (4.3.3)$$

$$\text{where } g(\mu) = \min \frac{1}{2} x^T D x + c^T x + \mu^T (A x - b) \\ \text{s.t. } 1 \leq x \leq u.$$

The structure of the primal problem allows us to express the function $g(\bullet)$ in a simpler form

$$g(\mu) = \min \frac{1}{2} \sum_{(i,j) \in E} D_{ij} x_{ij}^2 + \sum_{(i,j) \in E} (c_{ij} + \mu_i - \mu_j) x_{ij} - \sum_{i \in V} \mu_i b_i \\ \text{s.t. } 1 \leq x \leq u$$

where D_{ij} , for any $(i,j) \in E$, is a diagonal element of D .

Problem (4.3.3) is unconstrained. The two theorems below show that the objective function $g(\bullet)$ is concave and differentiable.

Theorem 4.3.1: The function $g(\mu)$ is concave.

Proof: The result follows because $g(\bullet)$ is the pointwise minimum of functions linear in μ . See Theorem 4.1.13 in Mangasarian [1969].

The following assumption and two lemmas are necessary for the proof of the second theorem.

Assumption 4.3.1: The set X , defined as $X = \{x : 1 \leq x \leq u\}$, is bounded.

Lemma 4.3.1: For any $\mu \in R^m$ the set $G(\mu)$, defined as $G(\mu) = \arg \min \{ \frac{1}{2} x^T D x + c^T x + \mu^T (A x - b) : x \in X \}$, is a singleton.

Proof: The result follows from the positive definiteness of D .

Lemma 4.3.2: Let $\{u^k\}$ be an infinite sequence of dual variables converging to \hat{u} , and let $x^k \in G(u^k)$ for all k . Then $\{x^k\}$ converges to \hat{x} where $\hat{x} \in G(\hat{u})$.

Proof: (By contradiction) Assume that there exists an $\epsilon > 0$ such that $\|x^k - \hat{x}\| > \epsilon$ for all k . Since X is compact, there exists a convergent subsequence, $\{x^k\}_K$, with limit x^* in X . Note that $\|x^* - \hat{x}\| > \epsilon$.

Furthermore, for each $k \in K$ we have

$$\frac{1}{2} x^{kT} D x^k + c^T x^k + u^{kT} (A x^k - b) \leq \frac{1}{2} \hat{x}^T D \hat{x} + c^T \hat{x} + u^{kT} (A \hat{x} - b).$$

Taking the limit as $k \in K$ approaches ∞ , $\{u^k\} \rightarrow \hat{u}$ and $\{x^k\} \rightarrow x^*$. It follows that

$$\frac{1}{2} x^{*T} D x^* + c^T x^* + \hat{u}^T (A x^* - b) \leq \frac{1}{2} \hat{x}^T D \hat{x} + c^T \hat{x} + \hat{u}^T (A \hat{x} - b).$$

Therefore, $x^* \in G(\hat{u})$ which implies that $G(\hat{u})$ is not a singleton, a contradiction to Lemma 4.3.1. ■

Theorem 4.3.2: The function $g(u)$ is differentiable.

Proof: Given $u_1, u_2 \in R^m$, $x_1 \in G(u_1)$ and $x_2 \in G(u_2)$. The following two inequalities hold from the definition of $g(\bullet)$:

$$\begin{aligned} g(u_1) - g(u_2) &\leq f(x_2) + u_1^T (A x_2 - b) - f(x_2) - u_2^T (A x_2 - b) \\ &= (u_1 - u_2)^T (A x_2 - b) \end{aligned}$$

$$\begin{aligned} g(u_2) - g(u_1) &\leq f(x_1) + u_2^T (A x_1 - b) - f(x_1) - u_1^T (A x_1 - b) \\ &= (u_2 - u_1)^T (A x_1 - b). \end{aligned}$$

These inequalities imply that

$$\begin{aligned}
 0 &\geq g(\mu_1) - g(\mu_2) - (\mu_1 - \mu_2)^T (Ax_2 - b) \\
 &\geq (\mu_1 - \mu_2)^T (Ax_1 - b) - (\mu_1 - \mu_2)^T (Ax_2 - b) \\
 &= (\mu_1 - \mu_2)^T A(x_1 - x_2) \geq -\|\mu_1 - \mu_2\| \|A(x_1 - x_2)\| \\
 0 &\geq \frac{g(\mu_1) - g(\mu_2) - (\mu_1 - \mu_2)^T (Ax_2 - b)}{\|\mu_1 - \mu_2\|} \geq -\|A(x_1 - x_2)\|.
 \end{aligned}$$

By Lemma 4.3.2, $x_1 \rightarrow x_2$ as $\mu_1 \rightarrow \mu_2$. Therefore

$$\lim_{\mu_1 \rightarrow \mu_2} \frac{g(\mu_1) - g(\mu_2) - (\mu_1 - \mu_2)^T (Ax_2 - b)}{\|\mu_1 - \mu_2\|} = 0.$$

Hence, $g(\mu)$ is differentiable. The gradient of $g(\bullet)$ at μ is

$$\delta g(\mu) = Ax - b.$$

Therefore, μ^* solves (4.3.3) if and only if $\delta g(\mu^*) = 0$, i.e.,

$$\delta g(\mu^*) = Ax^* - b = 0. \tag{4.3.4}$$

The next subsection presents a dual based iterative method for (4.3.3) that generates a convergent sequence of dual variables $\{\mu^k\}$ and a corresponding sequence of primal variables $\{x^k\}$ converging to a primal feasible solution.

4.3.1 Dual Ascent Method

In this subsection, we utilize the properties of $g(\bullet)$ to develop a general scheme to solve (4.3.3). This is based on the following procedure.

Dual Ascent Algorithm

Step 0: Let $\mu^0 = 0$ and set $k = 0$.

Step 1: Find an ascent direction d^k and go to 2.

Step 2: Let

$$\alpha^k \in \arg \max g(\mu^k + \alpha d^k).$$

Set $\mu^{k+1} = \mu^k + \alpha^k d^k$ and go to 3.

Step 3: If $\|\delta g(\mu^{k+1})\| \leq \epsilon$, terminate.

Otherwise, set $k = k + 1$ and go to 1.

4.3.1.1 Computation of d^k

Since $g(\bullet)$ is concave and differentiable, the direction d^k can be computed in terms of $\delta g(\mu^k)$, which includes the steepest ascent, quasi-Newton and CG methods. The method of steepest ascent usually works quite well during early iterations, but it performs poorly as the optimum is approached. Conventional quasi-Newton methods require too much storage to be practical for large networks. The only alternatives requiring only first derivatives and able to achieve rapid convergence are CG methods. Among the CG methods for nonquadratic objectives we have the following updating formulas:

i) Fletcher and Reeves [1964]:

$$d^{k+1} = \delta g(\mu^{k+1}) + \frac{\delta g(\mu^{k+1})^T \delta g(\mu^{k+1})}{\delta g(\mu^k)^T \delta g(\mu^k)} d^k$$

ii) Polak and Ribiere [1969]:

$$d^{k+1} = \delta g(u^{k+1}) + \frac{\delta g(u^{k+1})^T y^k}{\delta g(u^k)^T \delta g(u^k)} d^k$$

$$\text{where } y^k = \delta g(u^{k+1}) - \delta g(u^k).$$

4.3.1.2 Computation of α^k

We now propose a specific algorithm for solving the line search subproblem (Step 2) which can be stated as

$$\text{Find } \alpha^k \in \arg \max g(u^k + \alpha d^k) \quad (4.3.5)$$

$$\begin{aligned} \text{where } g(u^k + \alpha d^k) &= \min \quad \frac{1}{2} x^T D x + c^T x + (u^k + \alpha d^k)^T (A x - b) \\ \text{s.t. } &1 \leq x \leq u. \end{aligned} \quad (4.3.6)$$

Since $g(\bullet)$ is concave and differentiable, a necessary and sufficient optimality condition for α^k is

$$\delta g(u^k + \alpha^k d^k)^T d^k = (A x^k - b)^T d^k = 0 \quad (4.3.7)$$

where x^k is the solution of (4.3.6) at α^k .

Thus, imposing the condition (4.3.7), Problem (4.3.5) becomes

$$\begin{aligned} \min \quad &\frac{1}{2} x^T D x + c^T x + u^{kT} (A x - b) \\ \text{s.t. } \quad &d^{kT} (A x - b) = 0 \\ &1 \leq x \leq u \end{aligned} \quad (4.3.8)$$

and α^k is the optimal Lagrangian multiplier of the equality constraint. This problem transformation has also been discussed by Linn and Pang [1985] in the study of iterative methods for convex quadratic programs.

(4.3.8) can be rewritten as follows

$$\begin{aligned}
 \min \quad & \frac{1}{2} \sum_{(i,j) \in E} D_{ij} x_{ij}^2 + \sum_{(i,j) \in E} c_{ij}^1 x_{ij} + e' \\
 \text{s.t.} \quad & \sum_{(i,j) \in E} a_{ij} x_{ij} = b' \\
 & 1 \leq x \leq u
 \end{aligned} \tag{4.3.9}$$

where $c_{ij}^1 = c_{ij} + \mu_i^k - \mu_j^k$ for all $(i,j) \in E$

$$e' = \sum_{i \in V} \mu_i^k b_i$$

$$a_{ij} = d_i^k - d_j^k \quad \text{for all } (i,j) \in E$$

$$b' = \sum_{i \in V} d_i^k b_i.$$

(4.3.9) is a slightly more general problem than the one treated by Helgason et al. [1980], where the coefficients of the equality constraint are one and the lower bounds are zero. Below, their result is extended for this problem.

The central idea for solving (4.3.9) is to consider its optimality conditions and find the Lagrangian multiplier for its equality constraint. The KKT conditions are

$$D_{ij} x_{ij} + c_{ij}^1 + a_{ij} \alpha + v_{ij} - w_{ij} = 0 \quad \text{for all } (i,j) \in E \tag{4.3.10}$$

$$w_{ij}(1_{ij} - x_{ij}) = 0 \quad \text{for all } (i,j) \in E \tag{4.3.11}$$

$$v_{ij}(x_{ij} - u_{ij}) = 0 \quad \text{for all } (i,j) \in E \tag{4.3.12}$$

$$w, v \geq 0 \tag{4.3.13}$$

and x feasible.

The following solution as a function of α may be obtained

$$\begin{aligned} x_{ij}(\alpha) &= \max \left[\min \left\{ \frac{-c'_{ij} - a_{ij}\alpha}{D_{ij}}, u_{ij} \right\}, l_{ij} \right] \\ w_{ij}(\alpha) &= \max [D_{ij}l_{ij} + c'_{ij} + a_{ij}\alpha, 0] \\ v_{ij}(\alpha) &= \max [-D_{ij}u_{ij} - c'_{ij} - a_{ij}\alpha, 0]. \end{aligned} \quad (4.3.14)$$

Clearly, the above solution satisfies the bounding constraints, i.e.,

$$l \leq x \leq u$$

and $w, v \geq 0$.

The following three lemmas show that (4.3.10) - (4.3.12) are also satisfied.

Lemma 4.3.3: (4.3.14) satisfies (4.3.10).

Proof: For each $(i,j) \in E$, consider the following three cases:

- i) If $(-c'_{ij} - a_{ij}\alpha)/D_{ij} \geq u_{ij}$, then $x_{ij} = u_{ij}$,
 $-D_{ij}u_{ij} - c'_{ij} - a_{ij}\alpha \geq 0$, and $D_{ij}l_{ij} + c'_{ij} + a_{ij}\alpha \leq 0$.
 $-D_{ij}u_{ij} - c'_{ij} - a_{ij}\alpha \geq 0$ implies that $v_{ij} = -D_{ij}u_{ij} - c'_{ij} - a_{ij}\alpha$,
and $D_{ij}l_{ij} + c'_{ij} + a_{ij}\alpha \leq 0$ implies that $w_{ij} = 0$.
Then, $D_{ij}x_{ij} + c'_{ij} + a_{ij}\alpha + v_{ij} - w_{ij}$
 $= D_{ij}u_{ij} + c'_{ij} + a_{ij}\alpha - D_{ij}u_{ij} - c'_{ij} - a_{ij}\alpha - 0 = 0$.
- ii) If $l_{ij} < (-c'_{ij} - a_{ij}\alpha)/D_{ij} < u_{ij}$, then
 $x_{ij} = (-c'_{ij} - a_{ij}\alpha)/D_{ij}$, $-D_{ij}u_{ij} - c'_{ij} - a_{ij}\alpha < 0$,
and $D_{ij}l_{ij} + c'_{ij} + a_{ij}\alpha < 0$.
 $-D_{ij}u_{ij} - c'_{ij} - a_{ij}\alpha < 0$ implies that $v_{ij} = 0$, and
 $D_{ij}l_{ij} + c'_{ij} + a_{ij}\alpha < 0$ implies that $w_{ij} = 0$.

Then, $D_{ij}x_{ij} + c'_{ij} + a_{ij}\alpha + v_{ij} - w_{ij}$

$$= D_{ij}(-c'_{ij} - a_{ij}\alpha)/D_{ij} + c'_{ij} + a_{ij}\alpha + 0 - 0 = 0.$$

iii) If $(-c'_{ij} - a_{ij}\alpha)/D_{ij} \leq l_{ij}$, then $x_{ij} = l_{ij}$,

$D_{ij}l_{ij} + c'_{ij} + a_{ij}\alpha \geq 0$, and $-D_{ij}u_{ij} - c'_{ij} - a_{ij}\alpha \leq 0$.

$D_{ij}l_{ij} + c'_{ij} + a_{ij}\alpha \geq 0$ implies that $w_{ij} = D_{ij}l_{ij} + c'_{ij} + a_{ij}\alpha$, and

$-D_{ij}u_{ij} - c'_{ij} - a_{ij}\alpha \leq 0$ implies that $v_{ij} = 0$.

Then, $D_{ij}x_{ij} + c'_{ij} + a_{ij}\alpha + v_{ij} - w_{ij}$

$$= D_{ij}l_{ij} + c'_{ij} + a_{ij}\alpha + 0 - D_{ij}l_{ij} - c'_{ij} - a_{ij}\alpha = 0. \blacksquare$$

Lemma 4.3.4: (4.3.14) satisfies (4.3.11).

Proof: For each $(i,j) \in E$, consider the following two cases:

i) If $D_{ij}l_{ij} + c'_{ij} + a_{ij}\alpha \leq 0$, then $w_{ij} = 0$. This implies that

$$w_{ij}(l_{ij} - x_{ij}) = 0.$$

ii) If $D_{ij}l_{ij} + c'_{ij} + a_{ij}\alpha > 0$, then $l_{ij} > (-c'_{ij} - a_{ij}\alpha)/D_{ij}$. This implies that $x_{ij} = l_{ij}$ and, thus, $w_{ij}(l_{ij} - x_{ij}) = 0. \blacksquare$

Lemma 4.3.5: (4.3.14) satisfies (4.3.12).

Proof: For each $(i,j) \in E$, consider the following two cases:

i) If $-D_{ij}u_{ij} - c'_{ij} - a_{ij}\alpha \leq 0$, then $v_{ij} = 0$. This implies that

$$v_{ij}(x_{ij} - u_{ij}) = 0.$$

ii) If $-D_{ij}u_{ij} - c'_{ij} - a_{ij}\alpha > 0$, then $(-c'_{ij} - a_{ij}\alpha)/D_{ij} > u_{ij}$.

This implies that $x_{ij} = u_{ij}$ and, thus, $v_{ij}(x_{ij} - u_{ij}) = 0. \blacksquare$

Let $h(\bullet)$ be defined as

$$h(\alpha) = \sum_{(i,j) \in E} a_{ij}x_{ij}(\alpha)$$

$$= \sum_{(i,j) \in E} a_{ij} \max \left[\min \left\{ \frac{-c'_{ij} - a_{ij}\alpha}{D_{ij}}, u_{ij} \right\}, l_{ij} \right]$$

Hence, solving (4.3.9) is equivalent to finding α^k such that $h(\alpha^k) = b'$.

The theorem below is necessary for the convergence of the algorithm for α^k .

Theorem 4.3.3: The function $h(\alpha)$ is piecewise-linear and continuous nonincreasing.

Proof: Since for any $(i,j) \in E$, $x_{ij}(\bullet)$ is piecewise-linear and continuous, $h(\bullet)$ is also piecewise-linear and continuous.

In addition, if a_{ij} is zero or positive (negative), then $x_{ij}(\bullet)$ is nonincreasing (nondecreasing). Thus, $a_{ij}x_{ij}(\bullet)$ is always nonincreasing which implies that $h(\bullet)$ is nonincreasing. ■

For any $(i,j) \in E$, $x_{ij}(\bullet)$ has at most two breakpoints, i.e., $(-c'_{ij} - D_{ij}u_{ij})/a_{ij}$ and $(-c'_{ij} - D_{ij}l_{ij})/a_{ij}$. Therefore, the total number of different breakpoints of $h(\bullet)$, denoted as s , is less or equal than $2n$. Let $\alpha_1, \dots, \alpha_s$ be these breakpoints in increasing order. The algorithm below is based on the bisection method to obtain an interval defined by two consecutive breakpoints, (α_r, α_{r+1}) , containing α^k that is finally generated by interpolation.

Algorithm for α^k

Step 0: Let $p = 1$, $q = s$, $U = h(\alpha_p) = \sum_{(i,j) \in E} \max \{a_{ij}l_{ij}, a_{ij}u_{ij}\}$,
and $L = h(\alpha_q) = \sum_{(i,j) \in E} \min \{a_{ij}l_{ij}, a_{ij}u_{ij}\}$.

Step 1: (Bracketing)

If $q - p = 1$, go to 3.

Otherwise, set $r = \left\lfloor \frac{1}{2} (p + q) \right\rfloor$, where $\lfloor x \rfloor$ is the largest integer less or equal than x , and go to 2.

Step 2: Compute $h(\alpha_r)$ and consider the following three cases:

i) If $h(\alpha_r) = b'$, α_r is a solution and terminate.

ii) If $h(\alpha_r) < b'$, set $p = r$, $L = h(\alpha_r)$ and go to 1

iii) If $h(\alpha_r) > b'$, set $q = r$, $U = h(\alpha_r)$ and go to 1.

Step 3: (Interpolation)

$$\alpha = \alpha_p + \frac{(\alpha_q - \alpha_p)(b' - L)}{(U - L)}$$

α is a solution and terminate.

The work involved in the initialization of the algorithm in sorting the breakpoints of $h(\bullet)$ is not a hard task but may be time consuming since it has to be done at each iteration of the dual ascent method. However, as the method progresses, the dual vectors converge, so that the ordering of the breakpoints should stabilize, i.e., the order at a certain iteration should have a good chance of being the one needed for the next. Thus, the sorting algorithm employed to do this task can be enhanced by restarting from the old order.

Since $\alpha^k \geq 0$, another computational enhancement for the dual ascent method may be achieved by eliminating the breakpoints that are negative after modifying the values of p and U in Step 0 as follows

$$p = \arg \max \{ \alpha_i : \alpha_i \leq 0, i = 1, \dots, s \}$$

$$U = h(\alpha_p).$$

4.3.2 Examples

Table 4-4 lists the data of two small examples that have been used to test the dual ascent method. The first example has 4 nodes and 5 arcs, and the second 12 nodes and 22 arcs.

Tables 4-5 and 4-6 report the solution of these two problems using the steepest ascent and two CG methods to compute d^k . The CG methods for Example 1 restart at every 3 iterations, and for Example 2 at every 18 iterations.

Table 4-4: Data for Examples 1 and 2.

a) Data for Example 1:

Arcs					Nodes	
(i,j)	D_{ij}	c_{ij}	l_{ij}	u_{ij}	i	b_i
(1,2)	10.	1.	2.	8.	1	6.
(1,3)	2.	1.	0.	1.	2	0.
(2,3)	8.	2.	3.	5.	3	0.
(2,4)	2.	1.	0.	4.	4	-6.
(3,4)	2.	1.	0.	6.		

b) Data for Example 2:

Arcs					Nodes	
(i,j)	D_{ij}	c_{ij}	l_{ij}	u_{ij}	i	b_i
(1,3)	.8	1.	0.	11.	1	15.
(1,6)	1.0	4.	2.	8.	2	10.
(2,3)	.6	3.	0.	5.	3	0.
(2,4)	.2	7.	8.	9.	4	0.
(3,4)	.2	5.	0.	5.	5	0.
(3,5)	.4	2.	9.	11.	6	0.
(3,6)	.2	1.	0.	5.	7	0.
(4,6)	.8	9.	3.	7.	8	0.
(4,7)	.8	7.	0.	2.	9	0.
(5,7)	1.0	3.	0.	12.	10	0.
(5,8)	1.0	2.	5.	10.	11	-8.
(6,8)	.2	1.	0.	5.	12	-17.
(6,10)	.2	4.	2.	12.		
(7,9)	.4	5.	0.	10.		
(7,12)	.6	3.	0.	6.		
(8,9)	.8	8.	0.	1.		
(8,10)	.8	2.	0.	10.		
(8,11)	.6	4.	2.	6.		
(9,11)	.6	9.	2.	10.		
(10,9)	.6	7.	1.	5.		
(10,11)	.2	1.	0.	10.		
(10,12)	.4	13.	4.	15.		

Table 4-5: Solution of Example 1.

Method	Iter.	$g(\mu)$	$\ \delta g(\mu)\ $	% RE
Steepest Ascent	0	64.000	7.8740	68.0000
	1	108.393	6.1707	45.8037
	2	132.159	6.1467	33.9204
	3	159.318	3.2759	20.3408
	4	180.105	2.8462	9.9473
	5	187.621	2.2238	6.1897
	10	198.177	1.0934	0.9116
	20	199.948	0.1846	0.0260
	30	199.999	0.0312	0.0007
	33	200.000	0.0128	0.0002
Fletcher Reeves	0	64.000	7.8740	68.0000
	1	108.393	6.1707	45.8037
	2	174.992	2.4746	12.5041
	3	194.797	3.1537	2.6015
	4	198.228	0.6624	0.8858
	5	199.921	0.2906	0.0397
	6	200.000	0.0000	0.0000
Polak Ribiere	0	64.000	7.8740	68.0000
	1	108.393	6.1707	45.8037
	2	174.992	2.4746	12.5041
	3	194.797	3.1537	2.6015
	4	198.228	0.6624	0.8858
	5	199.921	0.2906	0.0397
	6	200.000	0.0000	0.0000

Table 4-6: Solution of Example 2.

Method	Iter.	$g(u)$	$\ \delta g(u) \ $	% RE
Steepest Ascent	0	259.000	22.5389	59.5086
	1	475.218	16.6512	25.7055
	2	506.759	19.7332	20.7745
	3	543.390	14.0195	15.0477
	5	565.842	10.6158	11.5376
	10	584.883	7.3529	8.5608
	20	610.801	7.1419	4.5088
	50	636.879	1.5225	0.4319
	100	639.403	0.5282	0.0373
	200	639.641	0.0291	0.0001
Fletcher Reeves	0	259.000	22.5389	59.5086
	1	475.218	16.6512	25.7055
	2	511.898	20.5915	19.9711
	3	536.549	20.7944	16.1172
	5	569.577	9.6002	10.9537
	10	612.532	8.4865	4.2382
	20	631.022	4.9819	1.3475
	30	639.322	1.5036	0.0499
	40	639.640	0.0736	0.0002
	43	639.641	0.0319	0.0000
Polak Ribiere	0	259.000	22.5389	59.5086
	1	475.218	16.6512	25.7055
	2	511.898	20.5915	19.9711
	3	536.232	20.8755	16.1668
	5	577.648	9.8241	9.6920
	10	623.873	5.9887	2.4652
	20	638.940	1.8017	0.1096
	30	639.639	0.1066	0.0003
	32	639.641	0.0427	0.0000

CHAPTER FIVE

SUMMARY

This dissertation has presented research directed towards the solution of models that can be formulated as optimization programs with a pseudoconvex objective function and network constraints. In practice, these models tend to be large, with many thousands of arcs (variables) and nodes (equality constraints). They are at least an order of magnitude larger than problems that can be handled routinely by standard state-of-the-art codes for linearly constrained nonlinear optimization and, therefore, it is necessary to develop algorithms that exploit the special structure of the model in order to be able to conserve computer memory requirements and to obtain an accurate solution in reasonable CPU time.

The restricted simplicial algorithm developed in Chapter Three is a general form of decomposition for linearly constrained nonlinear programs that blends first and second order methods. The linear subproblems for networks are solved by specialized algorithms that permit the practical solution of extremely large problems. It is not unusual to obtain solution times for the subproblems that are one hundredth of the time of general purpose LP codes. The nonlinear master problem, with nonnegativity constraints only, has small size (at most r

variables) and can be solved to high accuracy with two or three minor iterations of the projected Newton method for separable objective functions or the projected quasi-Newton for nonseparable objectives. The efficiency of the master problem relies on the fact that the initial solution is optimal with respect to the simplex defined by the extreme points generated in previous iterations. The finite convergence proof is a key result that characterizes the class of models that are especially suitable for this procedure, i.e., models for which the optimal solution belongs to a face of the feasible region with small dimension.

Separable strictly convex quadratic network problems are an important class of nonlinear networks for which the dual conjugate gradient based methods of Chapter Four are highly efficient. These methods take full advantage of the sparsity and structure of the constraint matrix. It has been shown that for the case of undirected arcs the dual approach, that reduces to solving a symmetric system of linear equations, outperforms the primal method of Chapter Three. When the flows have upper and lower bounds, the dual approach also looks promising for large problems, however, a complete computational study along with the analysis of preconditioning techniques to enhance the performance of the conjugate gradient methods employed, has not been done yet. Another important question that needs to be investigated is the extension of these methods to separable strictly convex functions by successive quadratic approximation at each iteration.

APPENDIX A

RSD FOR AN UNBOUNDED FEASIBLE REGION

Assumption 3.2.2 states that the feasible region of Problem (3.2.1), denoted as S , is bounded. Although this assumption is not too restrictive, it is still of interest to extend the RSD algorithm to the unbounded case.

If S is unbounded, it is possible that a solution may not exist. Thus, in order to guarantee the existence of a solution, Assumption 3.2.2 is replaced by the following:

Assumption A.1: For any direction $d \in \mathbb{R}^n$ such that $x + \alpha d \in S$ for all $\alpha \geq 0$, the following condition holds

$$\lim_{\alpha \rightarrow \infty} f(x + \alpha d) = \infty$$

In addition, since the linear subproblem may generate either an extreme point or an extreme direction, the feasible region of the master problem will be defined by the following class of generalized simplices [Rockafellar 1970].

Definition: Let $\{z_0, z_1, \dots, z_p\}$ be $p+1$ distinct points in \mathbb{R}^n and $\{u_1, \dots, u_q\}$ be q distinct directions in \mathbb{R}^n with $p+q \leq n$, where the vectors $z_1 - z_0, \dots, z_p - z_0, u_1, \dots, u_q$ are linearly independent. Then, the set

$$C = \{x : x = \sum_{i=0}^p \alpha_i z_i + \sum_{j=1}^q \beta_j u_j, \sum_{i=0}^p \alpha_i = 1, \alpha_i \geq 0, i = 0, \dots, p, \\ \beta_j \geq 0, j = 1, \dots, q\}$$

is called a $(p+q)$ -generalized simplex in R^n . In addition, since C is always contained in a manifold of dimension $p+q$, a $(p+q)$ -generalized simplex is said to have dimension $p+q$, written as $\dim C = p+q$.

The following version of RSD takes into account the extreme directions in the feasible region and, as in the basic algorithm, it is assumed that the total number of extreme points and extreme directions is restricted by a parameter r .

Step 0: Let x^0 be a feasible point of S . Set $W_S^0 = \emptyset$, $W_d^0 = \emptyset$, $W_x^0 = \{x^0\}$
 $k=0$.

Step 1: (Subproblem)

Solve

$$\min \{\delta f(x^k)y : y \in S\}.$$

If the subproblem generates an extreme point, y^k , such that $\delta f(x^k)(y^k - x^k) \geq 0$, or an extreme direction, d^k , such that $\delta f(x^k)d^k \geq 0$, x^k is a solution and terminate.

Otherwise,

i) If $|W_S^k| + |W_d^k| < r$, set $W_S^{k+1} = W_S^k \cup \{y^k\}$ or $W_d^{k+1} = W_d^k \cup \{d^k / \|d^k\|\}$, and $W_x^{k+1} = W_x^k$.

ii) If $|W_S^k| + |W_d^k| = r$, eliminate the element of W_S^k or W_d^k

with the minimal weight in the expression of x^k as a linear combination of $W^k \cup W_d^k$. Set $W_s^{k+1} = W_s^k \cup \{y^k\}$ or $W_d^{k+1} = W_d^k \cup \{d^k / \|d^k\|\}$, and $W_x^{k+1} = \{x^k\}$. Set $W^{k+1} = W_s^{k+1} \cup W_x^{k+1}$ and go to 2.

Step 2: (Master problem)

Let $x^{k+1} \in \arg \min \{f(x) : x \in H(W^{k+1}, W_d^{k+1})\}$

where $H(W^{k+1}, W_d^{k+1})$ is the generalized simplex defined by W^{k+1} and W_d^{k+1} .

Purge W_s^{k+1} , W_d^{k+1} and W_x^{k+1} of all the elements with zero weight in the expression of x^{k+1} as a linear combination of the elements of $W^{k+1} \cup W_d^{k+1}$. Set $k=k+1$ and go to 1.

The global and finite convergence for this version of RSD is similar to the one for the basic algorithm and is omitted.

APPENDIX B

RSD FOR NONLINEAR CONSTRAINTS

The RSD algorithm can be further extended to the case where some of the constraints are nonlinear. Consider the problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_j(x) \leq 0 \quad \text{for } j = 1, \dots, q \quad (\text{B.1}) \\ & A x \leq b \end{aligned}$$

Define the following sets:

$$S_1 = \{x : g_j(x) \leq 0, j = 1, \dots, q\}$$

$$S_2 = \{x : A x \leq b\}$$

$$S = S_1 \cap S_2$$

and consider the assumptions below, in addition to Assumptions 3.2.1 and 3.2.2.

Assumption B.1: $g_j(\bullet)$ is a continuously differentiable quasiconvex function for $j = 1, \dots, q$.

Assumption B.2: For each $x \in S$, the set $\{\nabla g_j(x) : g_j(x) = 0, j = 1, \dots, q\}$ is linearly independent.

Since the level sets of quasiconvex functions are convex sets, Assumption B.1 is necessary to guarantee that the feasible regions of the master problem are subsets of S . The second assumption avoids the

case of having nonlinear equality constraints in (B.1) since the possibility of replacing $g_j(x) = 0$ by $g_j(x) \leq 0$ and $-g_j(x) \leq 0$ is ruled out.

The algorithm described below is basically a generalization of the algorithm of Topkis and Veinott [1967] with the following improvements:

i) The line search step has been replaced by a minimization on a simplex defined by some boundary points as well as some interior points.

ii) The direction finding subproblem is a scaled version of the standard subproblem with respect to the set of nonlinear constraints.

iii) \bar{x} , the center of the box that defines the normalization constraints (i.e., $1^k \leq d \leq u^k$), only changes when the progress of the algorithm has been "satisfactory." This is necessary for the finite convergence proof.

Step 0: Let x^0 be a feasible point of S. Set $\bar{x} = x^0$, $1^0 = -\mu e$, $u^0 = \mu e$, $W_S^0 = \emptyset$, $W_x^0 = \{x^0\}$ and $k=0$, where $\mu > 0$ and $e = [1, \dots, 1]^T$.

Step 1: (Subproblem)

If $\|\delta f(x^k)\| = 0$, x^k is a solution and terminate.

Otherwise, let (z^k, d^k) be an optimal solution to the following subproblem:

$$\min \quad z$$

$$\text{s.t.} \quad \delta f(x^k)d - \|\delta f(x^k)\|z \leq 0$$

$$\delta g_j(x^k)d - \|\delta g_j(x^k)\|z \leq -g_j(x^k) \quad \text{for } j = 1, \dots, q$$

$$A d \leq b - A x^k$$

$$1^k \leq d \leq u^k$$

If $z^k \geq 0$, x^k is a solution and terminate.

Otherwise, let

$$\bar{\alpha} = \min \{1, \sup \{\alpha : g_j(x^k + \alpha d^k) \leq 0, j = 1, \dots, q\}\}$$

$$y^k = x^k + \bar{\alpha} d^k.$$

i) If $|W_S^k| < r$, set $W_S^{k+1} = W_S^k \cup \{y^k\}$ and $W_x^{k+1} = W_x^k$.

ii) If $|W_S^k| = r$, replace the element of W_S^k with the minimal weight in the expression of x^k as a convex combination of the elements of W^k with y^k to obtain W_S^{k+1} and let $W_x^{k+1} = \{x^k\}$.

Set $W^{k+1} = W_S^{k+1} \cup W_x^{k+1}$ and go to 2.

Step 2: (Master problem)

Let $x^{k+1} \in \arg \min \{f(x) : x \in H(W^{k+1})\}$.

Purge W_S^{k+1} and W_x^{k+1} of all elements with zero weight in the expression of x^{k+1} as a convex combination of the elements of W^{k+1} and go to 3.

Step 3: Define \bar{x} , l^{k+1} and u^{k+1} as follows:

i) If $\|\bar{x} - x^{k+1}\|_{\infty} \leq \mu/2$, set $l^{k+1} = -\mu e + (\bar{x} - x^{k+1})$ and $u^{k+1} = \mu e + (\bar{x} - x^{k+1})$.

ii) If $\|\bar{x} - x^{k+1}\|_{\infty} > \mu/2$, set $\bar{x} = x^{k+1}$, $l^{k+1} = -\mu e$ and $u^{k+1} = \mu e$.

Set $k = k+1$ and go to 1.

The global convergence of the above algorithm follows similar arguments to those of Topkis and Veinott [1967] and Zangwill [1969] for feasible direction methods.

A finite convergence result similar to Theorem 3.4.3 (Section 3.4) can be obtained under Assumption 3.4.1 and the following two:

Assumption B.3: x^* , the optimal solution of (B.1), belongs to the interior of the set S_1 .

Assumption B.4: μ , the parameter that defines the size of the normalization constraints, is small relative to ϵ , the radius of the largest open ball $B(x^*, \epsilon)$ around x^* in S_1 .

Under Assumptions B.3 and B.4 it can be shown that there exists a positive integer π , such that for all $k \geq \pi$ the following hold:

- i) \bar{x} does not change, i.e., Step 3(ii) is never executed.
- ii) If (z^k, d^k) is a solution of the subproblem in Step 1, then y^k is always equal to $x^k + d^k$ and is a solution of the following equivalent subproblem

$$\begin{aligned} \min \quad & \delta f(x^k)y \\ \text{s.t.} \quad & A y \leq b \\ & -\mu\epsilon + \bar{x} \leq y \leq \mu\epsilon + \bar{x}. \end{aligned} \tag{B.2}$$

Note that (B.2) is similar to the standard subproblem of RSD for linear constraints and its feasible region does not change with k , i.e., y^k is always an extreme point of a polytope defined by $\{y : Ay \leq b, -\mu\epsilon + \bar{x} \leq y \leq \mu\epsilon + \bar{x}\}$. In addition, this polytope contains x^* . Thus, since x^* can be obtained by minimizing $f(x)$ in the polytope, a pattern similar to Section 3.4 can be used for the finite convergence proof.

A preliminary computational experimentation of the above algorithm with large quadratically constrained quadratic programs has shown that the behavior of the algorithm is similar to the linearly constrained

case, i.e., the progress of the algorithm improves as r increases. This leads to the conclusion that the replacement of the line search step by a minimization on a simplex can improve the efficiency of some feasible direction methods, and make them useful and competitive for solving large nonlinear programming problems.

APPENDIX C

FURTHER COMPUTATIONAL NOTES

Computer Environment

The FORTRAN programs for the VAX 11/750 Computer (Subsections 3.7.1 and 4.3.2) were compiled using the FORTRAN 4.0 Compiler and executed under the VMS version 4.3 Operating System.

The FORTRAN programs for the IBM 3081D Computer (Subsections 3.7.2, 3.7.3, 3.7.4 and 4.2.2) were compiled using the FORTRAN H/Extended Compiler and executed under the OS/MVS Operating System.

Special Computer Science Techniques

For all of the network problems solved in this work, the network data was stored as standard linked lists. For details see Kennington and Helgason [1980].

Storage Requirements

The storage requirements for the RSD algorithm of Chapter 3 may be divided between requirements for the master problem and the subproblem. Since RSD may be integrated with any type of linear programming code, the subproblem requirements would depend on the problem type. The

principal arrays for the RSD master problem require $(r + 4)n + r^2 + 8r$ memory locations.

The principal arrays for the algorithms of Chapter 4 require $4n + 4m$ memory locations for the CG algorithm, $4n + 6m$ for the PCG algorithm, and $9n + 8m$ for the dual ascent method.

Parameters and Tolerances

A preliminary experimentation with the parameters and tolerances of the RSD master problem led to the following choices:

i) Stopping criteria: $RGAP_1 \leq \epsilon_1^k$, where $\epsilon_1^{k+1} = \min \{\epsilon_1^k, 0.1 * RGAP_2\}$, $RGAP_1$ is the relative gap for the master problem, $RGAP_2$ is the relative gap for the original problem, and $\epsilon_1^0 = 10^{-4}$.

ii) Armijo parameters: $\beta = 0.5$ and $\sigma = 10^{-4}$.

iii) Parameter defining the set of near binding constraints I_+^k : $\epsilon_2 = 10^{-2}$.

Availability of the codes

A tape containing the experimental codes in EBCDIC format is on file with the ISE departmental copy of the dissertation.

REFERENCES

- Abadie, J. and Carpenter, J. (1969): Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints. Optimization, R. Fletcher (Ed.), Academic Press, London, England, pp. 37-47.
- Axelsson, O. (1977): Solution of Linear Systems of Equations: Iterative Methods, in Sparse Matrix Techniques. Lecture Notes in Mathematics, Springer-Verlag, New York, pp. 1-51.
- Bachem, A. and Korte, B. (1977): Quadratic Programming over Transportation Polytopes. Report No. 7762-OR, Institut für Ökometrie und Operations Research, Bonn, West Germany.
- Balberg, I. and Binenbaum, N. (1983): Computer Study of the Percolation Threshold in a Two-Dimensional Anisotropic System of Conducting Sticks. Physical Review B, Vol. 28, No. 7, pp. 3799-3812.
- Balberg, I., Binenbaum, N. and Anderson, C.H. (1983): Critical Behavior of the Two-Dimensional Sticks System. Physical Review Letters, Vol. 51, No. 18, pp. 1605-1608.
- Bazaraa, M.S. and Shetty, C.M. (1979): Nonlinear Programming - Theory and Algorithms. John Wiley and Sons, New York.
- Beck, P., Lasdon, L. and Engquist, M. (1983): A Reduced Gradient Algorithm for Nonlinear Network Problems. ACM Transactions on Mathematical Software, Vol. 9, No. 1, pp. 57-70.
- Beckmann, M.J. (1951): Optimum Transportation on Networks. Cowles Commission Discussion Paper 2023.
- Bertsekas, D.P. (1976): On the Goldstein-Levitin-Polyak Gradient Projection Method. IEEE Transactions on Automatic Control, Vol. AC-21, No. 2, pp. 174-184.
- Bertsekas, D.P. (1979): Algorithms for Nonlinear Multicommodity Network Flow Problems. International Symposium on Systems Optimization and Analysis, Benseussan and Lions (Eds.), Springer-Verlag, New York, pp. 210-224.

- Bertsekas, D.P. (1982): Projected Newton Methods for Optimization Problems with Simple Constraints. SIAM Journal of Control and Optimization, Vol. 20, pp. 221-246.
- Bertsekas, D.P. and El Baz, D. (1984): Distributed Asynchronous Relaxation Methods for Convex Network Flow Problems. Report LIDS-P-1417, Laboratory for Information and Decision Systems, M.I.T., Cambridge, Massachusetts.
- Birkhoff, G. and Diaz, J.B. (1956): Nonlinear Network Problems. Quarterly of Applied Mathematics, Vol. 13, pp. 431-443.
- Bradley, G., Brown, G. and Graves G. (1977): Design and Implementation of Large Scale Primal Transshipment Algorithms. Management Science, Vol. 24, pp. 1-35.
- Cantor, D.G. and Gerla M. (1974): Optimal Routing in a Packet-Switched Computer Network. IEEE Transactions on Computers, Vol. C-23, No. 10, pp. 1062-1069.
- Charnes, A. and Lemke, C.E. (1954): Minimization of Nonlinear Separable Convex Functionals. Naval Research Logistics Quarterly, Vol. 1, No. 4, pp. 301-312.
- Collins, M., Cooper, L., Helgason, R., Kennington, J. and LeBlanc L. (1978): Solving the Pipe Network Analysis Problem Using Optimization Techniques. Management Science, Vol. 24, No. 7, pp. 747-760.
- Colville, A.R. (1968): A Comparative Study on Nonlinear Programming Codes. Technical Report No. 320-2949, IBM-Data Processing Division, New York Scientific Center, New York.
- Cooper, L. and Kennington, J. (1977): Steady State Analysis of Nonlinear Resistive Electrical Networks Using Optimization Techniques. Technical Report IEOR 77012, Southern Methodist University, Dallas, Texas.
- Cottle, R.W. and Duval, S.G. (1982): A Lagrangean Relaxation Algorithm for the Constrained Matrix Problem. Report SOL 82-10, Department of Operations Research, Stanford University, Stanford, California.
- Dafermos, S.C. and Sparrow, F.T. (1969): The Traffic Assignment Problem for a General Network. Journal of Research of the National Bureau of Standards B, Vol. 73B, No. 2, pp. 91-118.
- Dembo, R.S. (1979): Seminar at MIT, Cambridge, Massachusetts.

- Dembo, R.S. (1983): A Primal Truncated Newton Algorithm with Application to Large-Scale Nonlinear Network Optimization. Working Paper Series B No. 72, School of Organization and Management, Yale University, New Haven, Connecticut.
- Dembo, R.S. and Klincewicz, J.G. (1981): A Scaled Reduced Gradient Algorithm for Network Flow Problems with Convex Separable Costs. Mathematical Programming Study 15, pp. 124-147.
- Dembo, R.S. and Sahi, S. (1983): A Convergent Framework for Constrained Nonlinear Optimization. Working Paper Series B No. 69, School of Organization and Management, Yale University, New Haven, Connecticut.
- Dembo, R.S. and Tulowitzki, U. (1983a): On the Minimization of a Quadratic Function subject to Box Constraints. Working Paper Series B No. 71, School of Organization and Management, Yale University, New Haven, Connecticut.
- Dembo, R.S. and Tulowitzki, U. (1983b): Computing Equilibria on Large Multicommodity Networks: an Application of Truncated Quadratic Programming Algorithms. Working Paper Series B No. 65, School of Organization and Management, Yale University, New Haven, Connecticut.
- Dennis, J.E. and More, J.J. (1974): A Characterization of Superlinear Convergence and its Application to Quasi-Newton Methods. Mathematics of Computation, Vol. 28, pp. 549-560.
- Dennis, J.E. and More, J.J. (1977): Quasi-Newton Methods, Motivation and Theory. SIAM Review, Vol. 19, pp. 46-89.
- Fletcher, R. and Reeves, C. (1964): Function Minimization by Conjugate Gradients. Computer Journal, Vol. 7, pp. 149-154.
- Florian, M., Guelat, J. and Spiess, H. (1983): An Efficient Implementation of the Partan Variant of the Frank-Wolfe Algorithm for Equilibrium Traffic Assignment. ORSA/TIMS Joint National Meeting, Orlando, Florida.
- Frank, M. and Wolfe, P. (1956): An Algorithm for Quadratic Programming. Naval Research Logistics Quarterly, Vol. 3, pp. 99-110.
- Geoffrion, A.M. (1970): Elements of Large-Scale Mathematical Programming. Management Science, Vol. 21, No. 11, pp. 652-691.
- Gill, P.E., Murray, W. and Wright, M.H. (1981): Practical Optimization. Academic Press, London, England.

- Gill, P.E., Murray, W., Saunders, M.A. and Wright, M.H. (1983): User's Guide for SOL/NPSOL: A Fortran Package for Nonlinear Programming. Report SOL 83-12, Department of Operations Research, Stanford University, Stanford, California.
- Grigoriadis, M.D. and Hsu, T. (1979): The Rutgers Minimum Cost Network Flow Subroutine. SIGMAP Bulletin, pp. 17-18.
- Guelat, J. (1983): Algorithmes pour le Probleme d'Affectation du Traffic d'Equilibre avec Demandes Fixes: Comparaisons. Publication 299, Centre de Recherche sur les Transports, Universite de Montreal, Montreal, Canada.
- Hanscom, M., Lafond, L., Lasdon, L. and Pronovost, G. (1978): Modeling and Resolution of the Medium Term Energy Generation Planning Problem for a Large Hydroelectric System. Management Science, Vol. 26, No. 7, pp. 659-668.
- Hearn, D.W. (1982): The Gap Function of a Convex Program. Operations Research Letters, Vol. 1, pp. 67-71.
- Helgason, R.V., Kennington, J.L. and Lall, H. (1980): A Polynomially Bounded Algorithm for a Single Constrained Quadratic Program. Mathematical Programming, Vol. 18, pp. 338-343.
- Hestenes, M.R. (1980): Conjugate Direction Methods in Optimization. Springer-Verlag, New York.
- Hestenes, M.R. and Stiefel, E.L. (1952): Methods of Conjugate Gradient for Solving Linear Systems. Journal of Research of the National Bureau of Standards, Vol. 49, pp. 409-436.
- Himmelblau, D.M. (1972): Applied Nonlinear Programming. McGraw-Hill, New York.
- Hollaway, C.A. (1974): An Extension of the Frank-Wolfe Method of Feasible Directions. Mathematical Programming, Vol. 6, No. 1, pp. 14-27.
- Jacoby, S. and Kowalik, J. (1980): Mathematical Modeling with Computers. Prentice-Hall, Englewood Cliffs, New Jersey.
- Jennings, A. (1977): Influence of the Eigenvalue Spectrum on the Convergence Rate of the Conjugate Gradient Method. Journal of the Institute of Mathematics and Its Applications, Vol. 20, pp. 61-72.
- Kamesam, P.V. and Meyer, R.R. (1984): Multipoint Methods for Nonlinear Networks. Mathematical Programming Study 22, pp. 185-205.
- Kennington, J.L. (1984): Private Communication.

- Kennington, J.L. and Helgason, R.V. (1980): Algorithms for Network Programming. John Wiley and Sons, New York.
- Kershaw, D.S. (1978): The Incomplete Cholesky-Conjugate Gradient Method for the Iterative Solution of Systems of Linear Equations. Journal of Computational Physics, Vol. 26, pp. 43-65.
- Klincewicz, J.G. (1983): A Newton Method for Convex Separable Network Flow Problems. Networks, Vol. 13, No. 3, pp. 427-442.
- Lawphongpanich, S. and Hearn, D.W. (1983): Restricted Simplicial Decomposition with Application to the Traffic Assignment Problem. Research Report No. 83-8, Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida.
- LeBlanc, L.J. (1976): The Conjugate Gradient Technique for Certain Quadratic Network Problems. Naval Research Logistics Quarterly, Vol. 23, pp. 597-602.
- LeBlanc, L.J., Helgason, R.V. and Boyce, D.E. (1985): Improved Efficiency of the Frank-Wolfe Algorithm for Convex Network Programs. Transportation Science, Vol. 19, No. 4, pp. 445-462.
- LeBlanc, L.J., Morlok, E.K. and Pierskalla, W.P. (1975): An Efficient Approach to Solving the Road Network Equilibrium Traffic Assignment Problem. Transportation Research, Vol. 9, pp. 309-318.
- Linn, Y.Y. and Pang, J.S. (1985): Iterative Methods for Large Convex Quadratic Programs: a Survey. Working Paper, School of Management, University of Texas at Dallas, Richardson, Texas.
- Luenberger, D.G. (1984): Linear and Nonlinear Programming. Addison-Wesley, Reading, Massachusetts.
- Mangasarian, O. (1969): Nonlinear Programming. McGraw-Hill, New York.
- Mulvey, J.M. (1985): Nonlinear Network Models in Finance. Advances in Mathematical Programming and Financial Planning, JAI Press, Greenwich, Connecticut.
- Munksgard, N. (1980): Solving Sparse Symmetric Sets of Linear Equations by Preconditioned Conjugate Gradients. ACM Transactions on Mathematical Software, Vol. 6, No. 2, pp. 206-219.
- Murtaugh, B. and Saunders, M. (1978): Large-Scale Linearly Constrained Optimization. Mathematical Programming, Vol. 15, pp. 291-314.

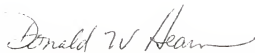
- Nickel, R.H. and Tolle, J.W. (1984): A Sequential Quadratic Programming Algorithm for Solving Large Sparse Nonlinear Programs. Professional Paper 416, Center for Naval Analyses, Alexandria, Virginia.
- Nguyen, S. (1974): An Algorithm for the Traffic Assignment Problem. Transportation Science, Vol. 8, pp. 203-216.
- Nguyen, S. (1976): A Unified Approach to Equilibrium Methods for Traffic Assignment. Traffic Equilibrium Methods, M.A. Florian (Ed.), Springer-Verlag, New York, pp. 148-182.
- Nguyen, S. and James, L. (1975): TRAFFIC - An Equilibrium Traffic Assignment Program. Report 17, Centre de Recherche sur les Transport, Universite de Montreal, Montreal, Canada.
- O'Leary, D.P. (1980): A Generalized Conjugate Gradient Algorithm for Solving a Class of Quadratic Programming Problems. Linear Algebra and Its Applications, Vol. 34, pp. 371-399.
- Polak, E. (1971): Computational Methods in Optimization: A Unified Approach. Academic Press, New York.
- Polak, E. and Ribiere, G. (1969): Note sur la Convergence de Methodes de Directions Conjugues. Revue Francaise Informat Recherche Operationnelle, Vol. 16, pp. 35-43.
- Powell, M.J.D. (1978): A Fast Algorithm for Nonlinear Constrained Calculations, in Numerical Analysis. Lecture Notes in Mathematics, No. 630, Springer-Verlag, New York.
- Rockafellar, R.T. (1970): Convex Analysis. Princeton University Press, Princeton, New Jersey.
- Rockafellar, R.T. (1984): Network Flows and Monotropic Programming. John Wiley, New York.
- Rosenthal, R.E. (1981): A Nonlinear Network Flow Algorithm for Maximization of Benefits in a Hydroelectric Power System. Operations Research, Vol. 29, No. 4, pp. 763-786.
- Schittkowski, K. (1984): NLPQL: A FORTRAN Subroutine for Solving Constrained Nonlinear Programming Problems. Report, Institut für Informatik, Universität Stuttgart, Stuttgart, West Germany.
- Shanno, D.F. (1978): Conjugate Gradient Methods with Inexact Searches. Mathematics of Operations Research, Vol. 3, No. 3, pp. 244-256.
- Steenbrink, P.A. (1974): Optimization of Transport Networks. John Wiley and Sons, New York.

- Syslo, M.M., Deo, N. and Kowalik, S. (1983): Discrete Optimization Algorithms with Pascal Programs. Prentice-Hall, New York.
- Topkis, D.M. and Veinott, A. Jr. (1967): On the Convergence of some Feasible Direction Algorithms for Nonlinear Programming. SIAM Journal on Control, Vol. 5, No. 2, pp. 268-279.
- von Hohenbalken, B. (1975): A Finite Algorithm to Maximize Certain Pseudoconcave Functions on Polytopes. Mathematical Programming, Vol. 8, pp. 189-206.
- von Hohenbalken, B. (1977): Simplicial Decomposition in Nonlinear Programming Algorithms. Mathematical Programming, Vol. 13, pp. 49-68.
- Wardrop, J. (1952): Some Theoretical Aspects of Road Traffic Research. Proceedings of the Institute of Civil Engineers, Vol. 1, Part II, pp. 325-378.
- Wolfe, P. (1961): A Duality Theorem for Nonlinear Programming. Quarterly of Applied Mathematics, Vol. 19, No. 3, pp. 239-244.
- Wolfe, P. (1967): Methods of Nonlinear Programming. Nonlinear Programming, J. Abadie (Ed.), Academic Press, London, England, pp. 37-47.
- Wolfe, P. (1970): Convergence Theory in Nonlinear Programming. Integer and Nonlinear Programming, J. Abadie (Ed.), North Holland, New York.
- Yang, E.K. and Tolle, J.W. (1985): A Class of Methods for Solving Large Convex Quadratic Programs subject to Box Constraints. Working Paper, Management Science Department, University of Massachusetts, Boston, Massachusetts.
- Zangwill, W.I. (1967): The Convex Simplex Method. Management Science, Vol. 14, pp. 221-283.
- Zangwill, W.I. (1969): Nonlinear Programming: A Unified Approach. Prentice Hall, Englewood Cliffs, New Jersey.
- Zenios, S.A. and Mulvey, J.M. (1986): Relaxation Techniques for Strictly Convex Network Problems. Annals of Operations Research, Volume on Algorithms and Software for Optimization, to appear.

BIOGRAPHICAL SKETCH

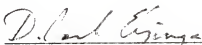
Jose A. Ventura was born on November 3, 1954, in Vilafranca del Penedes, Spain. In 1972, he attended the Polytechnical University of Barcelona, Spain, where he received a Bachelor of Science degree in industrial engineering in October, 1979. He married Marta Jaen in July, 1981. In 1982, he enrolled in the Industrial and Systems Engineering Department of the University of Florida, where he was awarded a Master of Engineering degree with specialization in operations research in August, 1984.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.




Donald W. Hearn, Chairman
Professor of Industrial and Systems
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



D. Jack Elzinga, Co-Chairman
Professor of Industrial and Systems
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Harold P. Benson
Associate Professor of Management

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Richard L. Francis
Professor of Industrial and Systems
Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Horst W. Hamacher

Horst W. Hamacher
Associate Professor of Industrial
and Systems Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August, 1986.

Herbert A. Davis

Dean, College of Engineering

Dean, Graduate School